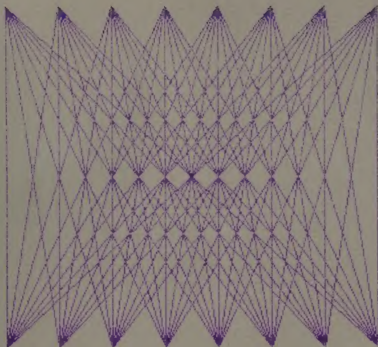
40 grafische programma's voor de ZX Spectrum

Leer programmeren
met hoge resolutie graphics
in BASIC

M. Sutter



ACADEMIC SERVICE

6

40 GRAFISCHE PROGRAMMA's
voor de ZX Spectrum

In deze reeks zijn verschenen:

40 grafische programma's voor de Commodore 64
40 grafische programma's in MSX BASIC
40 grafische programma's voor de Apple II, IIe en IIc
40 grafische programma's in IBM- en GW-BASIC
40 grafische programma's voor de ZX Spectrum
40 grafische programma's voor de Electron en BBC

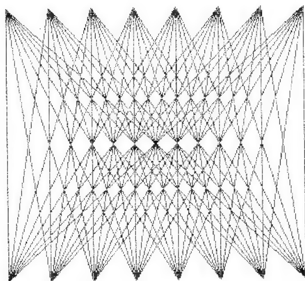
Voor de ZX-Spectrum is verschenen:

Ontdek de ZX-Spectrum - *Tim Hartnell*

40 grafische programma's voor de ZX Spectrum

Leer programmeren
met hoge resolutie graphics
in BASIC

M. Sutter



ACADEMIC SERVICE

Oorspronkelijke titel: *Programmieren mit hochauflösender Grafik*
Verschenen bij: Mikro+Kleincomputer Informa Verlag AG, Luzern

Vertaling en bewerking: Nok van Veen

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Sutter, Marcel

40 grafische programma's voor de ZX Spectrum / Marcel Sutter ; [vert. uit het Duits door Nok van Veen]. - Den Haag : Academic Service. - Ill.
Vert. van: Programmieren mit hochauflösender Grafik. - Luzern : Mikro+Kleincomputerverlag, 1984.
ISBN 90-6233-180-7
SISO 365.3 UDC 681.3.06 UGI 650
Trefw.: computerprogramma's / ZX Spectrum (computer).

Uitgegeven door: Academic Service
Postbus 96996
2509 JJ Den Haag
Zetwerk: multitASk, Blaricum
Druk: Krips Repro Meppel
Bindwerk: Meeuwis, Amsterdam
Omslagontwerp: Leo Bolt
ISBN 90 6233 180 7

Voorwoord

Door het toenemende gebruik van microcomputers maken de grafische toepassingen een sterke ontwikkeling door. Het beeldscherm is tegenwoordig, veel meer dan de printer, het afdrukapparaat bij uitstek. Grafische toepassingen (graphics) vinden we bij computerspelletjes, bij computerkunst en in toenemende mate op de werkplek van ingenieurs, ontwerpers, architecten en anderen die zich bezighouden met Computer Ondersteund Ontwerpen (Computer Aided Design).

De geïnteresseerde leek is vaak verrukt van de mooie 'plaatjes', die bij demonstraties getoond worden, en denkt dat hiervoor hele ingewikkelde programma's nodig zijn. Iemand, die iets van wiskunde afweet, weet hoe een functie eruitziet en kan doorgaans zonder veel moeite grafische programma's maken.

Ik heb voor het Zwitserse Computertijdschrift 'MIKRO+KLEINCOMPUTER' een cursus 'programmeren met hoge resolutie' geschreven, die in een aantal afleveringen van het blad is verschenen. Al direct na het verschijnen van het eerste artikel werd ik werkelijk overstelpt met enthousiaste reacties van de lezers. Dit heeft ons aangezet tot het maken van dit boek.

Veel van de grafische toepassingsprogramma's worden geschreven voor een bepaald merk en type microcomputer of voor een bepaalde plotter. Wie een dergelijk programma wil herschrijven voor een ander merk microcomputer zal veel moeilijkheden ondervinden en vaak zelfs zijn pogingen staken.

De veertig programma's in dit boek zijn geschreven in standaard BASIC en gebruiken geen POKE- en PEEK-opdrachten. Bovendien worden slechts twee grafische opdrachten gebruikt, te weten het inschakelen van de hoge-resolutiestand en het verbinden van twee punten door een rechte lijn. De illustraties die als voorbeeld dienen van hetgeen de programma's tekenen zijn gemaakt op de miniplotter van de Sharp PC-1500.

Wij hopen dat de lezers veel plezier aan deze programma's zullen beleven en dat zij voor velen een aanmoediging zullen zijn om het terrein van de computergraphics verder te verkennen. Mijn dank gaat uit naar de heer H.J. Ottenbacher van Micro+Kleincomputer; zonder hem zou dit boek nooit het daglicht hebben gezien.

Marcel Sutter
voorjaar 1984

Voorwoord bij de Nederlandse uitgave

De auteur, Marcel Sutter, heeft dit boek oorspronkelijk geschreven in standaard Microsoft BASIC. In zijn boek is een aantal appendices opgenomen met programmalistings voor een aantal microcomputers, zoals de Commodore 64, Apple II. Wij hebben besloten dit boek voor een zestal microcomputers afzonderlijk uit te geven. Het boek dat u nu leest is de ZX Spectrum versie. Alle programma's uit dit boek zijn getest en hebben gedraaid op een ZX Spectrum. De programma's zijn vanuit het Spectrum-geheugen op een printer afgedrukt en zo in het boek opgenomen. Tik ze dan ook precies zo in als ze in het boek zijn afgedrukt.

Wij raden u aan al bij de eerste programma's te experimenteren met een aantal grafische mogelijkheden van de ZX Spectrum. De programma's gebruiken allemaal een zwart potlood (INK) op wit papier (PAPER), waarmee de zwart-op-wit tekeningen gemaakt worden. Probeer gerust andere combinaties door de opdrachten INK en PAPER te gebruiken. Met FLASH en BRIGHT kunnen ook leuke effecten bereikt worden. In de appendix vindt u een BASIC-programma voor het inlezen van een machinetaalroutine, waarmee vlakken gekleurd kunnen worden.

Wilt u op uw printer laten afdrukken wat een programma getekend heeft, zet dan vlak voor de STOP-opdracht de opdracht COPY in het programma. Hierdoor wordt het schermbeeld op de printer afgedrukt.

Het aantal beeldpunten (pixels) dat de ZX Spectrum gebruikt is 256 horizontaal (genummerd 0 t/m 255) en 176 (genummerd 0 t/m 175) vertikaal. Veel illustraties in dit boek zijn gemaakt door de auteur op een plotter met 220 beeldpunten horizontaal en vertikaal. Zij geven dus wel een goede indruk van wat u op het scherm kunt verwachten, maar zijn niet precies gelijk aan hetgeen u op uw ZX Spectrum scherm zult zien.

Alle programma's zijn kort. Verfraai ze, verbeter ze en breid ze uit. De mogelijkheden zijn enorm. Omdat de programma's opzettelijk klein

gehouden zijn, wordt er bijvoorbeeld niet getest of bepaalde waarden, die u moet intoetsen, het programma door een of andere fout laten afbreken. Een verbetering zou zijn om deze tests wel op te nemen, waardoor een programma alleen met de juiste invoer aan het werk gaat. De appendix bevat een aantal suggesties voor het verfraaien en uitbreiden van de programma's.

Het is eigenlijk een boek voor elke ZX Spectrum bezitter. Vindt u de meestal wat wiskundige uitleg bij de programma's teveel van het goede, tik de programma's dan gewoon in en draai ze; u zult verrukt zijn van het resultaat. Kunt u zich nog iets van de sinus en cosinus herinneren of zit u nog op school dan zult u weinig moeite hebben met de theorie in dit boek. Voor leraren en leerlingen zijn wellicht de programma's voor het tekenen van functies interessant. Het boek behandelt ook het principe van driedimensionaal tekenen met verborgen lijnen (hidden lines). Erg leuk zijn de vijf programma's waarmee getekend kan worden zoals dat in de taal LOGO kan. Dit geeft zeer fraaie tekeningen, vooral de Turtle-graphics. Ook bevat het boek een vijftal educatieve toepassingsprogramma's, waarin graphics de hoofdrol spelen.

Tot slot willen wij Nico Huffels bedanken voor de assistentie bij het bewerken van de programma's in dit boek.

Nok van Veen
augustus 1985

Inhoud

1 ZX Spectrum Graphics

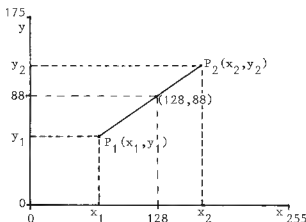
Het beeldscherm van de ZX Spectrum is als grafisch scherm te gebruiken, waarbij horizontaal 256 en vertikaal 176 puntjes (pixels) aan of uit gezet kunnen worden. Op dit grafische scherm kunnen we zogenaamde hoge resolutie graphics tekenen met behulp van de POINT-, PLOT-, DRAW- en CIRCLE-opdracht.

In alle programma's gaan we uit van de standaardinstelling voor de achtergrondkleur (PAPER) en de afdrukkleur (INK), de papier- en schrijfkleur. De standaardinstelling is wit voor PAPER en zwart voor INK. We tekenen dus met een zwart potlood op wit papier. U kunt echter bij het tekenen gebruik maken van acht kleuren (zie pagina 22). Experimenteer in uw programma's met PAPER en INK en kijk wat het effect ervan is. Ook kunt u met BRIGHT en FLASH beelden helderder respectievelijk flikkerend maken. In de appendix staat een programma met een PAINT-routine voor het kleuren (vullen) van vlakken. Dit kunt u bij de programma's in dit boek gebruiken.

De PLOT-opdracht wordt gebruikt om aan te geven waar het tekenen van een lijn moet beginnen. Met de DRAW-opdracht wordt vervolgens aangegeven naar welk punt de lijn getrokken moet worden. De oorsprong van het grafische beeldschermcoördinatenstelsel ligt bij de ZX Spectrum in de linkerbenedenhoek. De x-as wijst horizontaal naar rechts en de y-as wijst vertikaal naar boven. De 256 beeldpunten op de x-as worden genummerd van 0 tot en met 255 en de 176 puntjes op de y-as van 0 tot en met 175. Een punt op het beeldscherm (een pixel) kunnen we dan aangeven met een x-coördinaat en een y-coördinaat, bijvoorbeeld het punt (128,88), dat in het midden van het scherm ligt (zie p.2).

Om de ZX Spectrum een lijn te laten trekken tussen de punten P_1 en P_2, met respectievelijk coördinaten (x_1,y_1) en (x_2,y_2) geven we de volgende opdrachten:

```
PLOT x1,y1
DRAW x2-x1,y2-y1
```



Als (x_1,y_1) het punt (55,70) is en (x_2,y_2) het punt (150,135), dan worden deze opdrachten:

```
PLOT 55,70
DRAW 95,65
```

De lijn wordt dan vanuit P_1 naar P_2 getrokken. Willen we vanuit P_2 naar P_1 een lijn trekken. dan nemen we

```
PLOT x2,y2
DRAW x1-x2,y1-y2
```

wat voor ons voorbeeld neerkomt op:

```
PLOT 150,135
DRAW -95,-65
```

We zouden de lijn ook kunnen tekenen door een heleboel puntjes tussen P_1 en P_2, die precies op de verbindingslijn van P_1 en P_2 liggen, te laten oplichten. Deze methode staat bekend als 'puntgraphics'. Wij gebruiken echter in dit boek de 'lijngraphics'-methode, waarbij we steeds twee punten, die natuurlijk best vlak naast elkaar kunnen liggen, door een rechte lijn (of lijntje) met elkaar verbinden. Hierdoor hoeven we alleen de coördinaten van de uiteinden van de te tekenen lijn te berekenen en bovendien zien de tekeningen er beter uit dan met de puntgraphics-methode.

Bewust hebben we ons tot het gebruik van PLOT en DRAW beperkt en ook tot het gebruik van 'zwart op wit'. Mede hierdoor zijn de programma's kort en overzichtelijk, zodat de tekentechniek goed tot uitdrukking komt. Wij raden u echter aan de programma's uit te breiden met meer grafische opdrachten en kleur. In de appendix geven we hiervan enkele voorbeelden. Ook geven we in de appen-

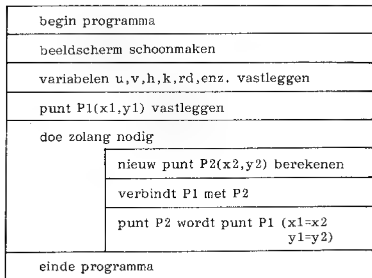
dix een programma voor het inlezen van een machinetaalroutine, waarmee een heel vlak gekleurd kan worden. Zo hier en daar in het boek vindt u suggesties voor het aanbrengen van kleur in de tekeningen.

Structuur van een grafisch programma

In alle programma's gebruiken we steeds dezelfde variabelen en dezelfde programmastructuur. De verschillende variabelen betekenen:

x1,y1	coördinaten van het beginpunt van een lijn
x2,y2	coördinaten van het eindpunt van een lijn
u,v	oorsprong van het wiskundige coördinatenstelsel; dikwijls is dit het midden van het beeldscherm (128,88)
h	de waarde 0,5 voor het afronden op helen
k	de vermenigvuldigingsfactor voor functiewaarden
w,w1	hoeken bij trigonometrische functies
rd	het getal π/180 voor het omrekenen van graden in radialen

Een grof structuurdiagram voor de programma's is hieronder weergegeven.

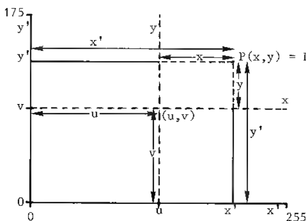


In alle programma's gebruiken we twee belangrijke transformatie-formules.

Voor het grafische coördinatenstelsel van de ZX Spectrum is de linkerbenedenhoek van het scherm de oorsprong. Wijzelf gebruiken echter bijna altijd een coördinatenstelsel met de oorsprong in het midden van het scherm. Om de beeldschermcoördinaten x',y' (zoals de ZX Spectrum ze gebruikt) te berekenen uit de coördinaten x,y zoals wij ze gebruiken hanteren we de transformatieformules:

x' = INT(u+x+h) : y' = INT(v+y+h)

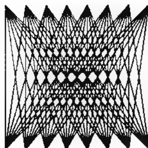
Bekijk de figuur hieronder eens. Duidelijk is dat elk punt p(x,y) in ons coördinatenstelsel (gestippeld assenkruis) door bovenstaande formules wordt getransformeerd (overgebracht) naar een punt p(x',y') in het coördinatenstelsel van de ZX Spectrum. Hierbij hoeft onze oorsprong (u,v) natuurlijk niet altijd precies in het midden van het beeld te liggen.



Genoeg theorie, nu de programma's. De ideeën voor de programma's hebben wij gekregen bij het doorbladeren van Amerikaanse, Duitse en Franse computertijdschriften. Alle programma's zijn echter eigen creaties of bewerkingen en vereenvoudigingen van reeds gepubliceerde programma's.

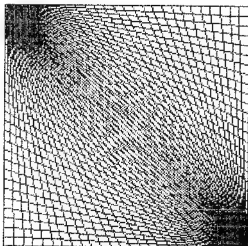
Programma 1 tekent een 'diagonaalweb'. Elk van de acht bovenste punten wordt door een rechte lijn verbonden met elk van de onderste acht punten.

```
100 REM programma 1
       diagonaalweb
110 CLS
120 LET y1=0 : LET y2=175
130 FOR a=0 TO 255 STEP 36
140  FOR b=0 TO 255 STEP 36
150   PLOT a,y1
160   DRAW b-a,y2
170  NEXT b
180 NEXT a
190 IF INKEY$="" THEN GO TO 190
200 STOP
```

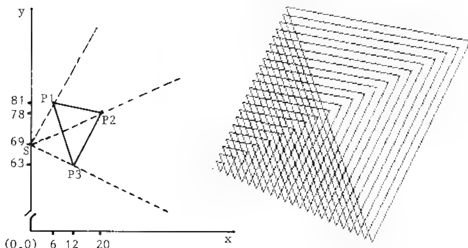


Programma 2 is een fraai voorbeeld van het Moiree-effect. Geniet ervan.

```
100 REM programma 2
        Moiree effect
110 CLS
120 FOR j=0 TO 175 STEP 7
130  PLOT 40,175
140  DRAW j,-175
150  PLOT 40,175
160  DRAW 175,j-175
170 NEXT j
180 FOR j=0 TO 175 STEP 7
190  PLOT 215,0
200  DRAW -175,j
210  PLOT 215,0
220  DRAW -175+j,175
230 NEXT j
240 IF INKEY$="" THEN GO TO 240
250 STOP
```



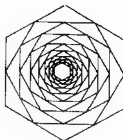
Programma 3 tekent een reeks driehoeken in perspectief. Het 'centrum van vermenigvuldiging' heeft de coördinaten S(0,69). De drie hoekpunten van de driehoek die vermenigvuldigd wordt zijn P1(6,81), P2(20,78) en P3(12,63). Alle coördinaten zijn beeldschermcoördinaten (oorsprong links onderaan). De drie richtingen van vermenigvuldiging (richtingsvectoren) zijn SP_1, SP_2 en SP_3. Deze worden in het programma als a(1);b(1), a(2);b(2) en a(3);b(3) vastgelegd. De vermenigvuldigingsfactor k is de lusvariabele van de buitenste FOR-NEXT-lus. k loopt van 0 tot 9 in stappen van 0,4. Hieronder zien we de uitgangssituatie, het resultaat van het programma en het programma zelf.




```
100 REM programma 3 driehoeken
        in perspectief
110 CLS
120 DIM x(3) : DIM y(3)
130 DIM a(3) : DIM b(3)
140 FOR j=1 TO 3
150  READ a(j),b(j)
160 NEXT j
170 LET x0=0 : LET y0=69
180 FOR k=0 TO 9 STEP 0.4
190  FOR j=1 TO 3
200   LET x(j)=x0+k*a(j)
210   LET y(j)=y0+k*b(j)
220  NEXT j
230  PLOT x(1),y(1)
240  DRAW x(2)-x(1),y(2)-y(1)
250  PLOT x(2),y(2)
260  DRAW x(3)-x(2),y(3)-y(2)
270  PLOT x(3),y(3)
280  DRAW x(1)-x(3),y(1)-y(3)
290 NEXT k
300 IF INKEY$="" THEN GO TO 300
310 STOP
320 DATA 6,12,20,9,12,-6
```

Programma 4 tekent een reeks ingeschreven regelmatige zeshoeken. Behalve bij de eerste zeshoek, liggen alle zes de hoeken van elke zeshoek precies in het midden van een zijde van de omgeschreven zeshoek. De coördinaten van de hoekpunten van de eerste (buitenste) zeshoek worden met trigoniometrische functies (sinus- en cosinusfunctie) berekend. De hoek die hierbij als argument van de functies gebruikt wordt doorloopt de waarden 60°, 120°, 180°, 240°, 300°, 360° en 420°. De middelpunten van de zijden van een zeshoek worden aangegeven met a(1);b(1) t/m a(6);b(6). Dit worden de hoekpunten van de volgende zeshoek.

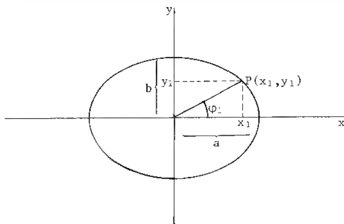
```
 100 REM programma 4
          ingeschreven zeshoeken
 110 CLS
 120 DIM x(7) : DIM y(7)
 130 DIM a(7) : DIM b(7)
 140 LET u=128 : LET v=88
 150 LET r=88 : LET h=0.5
 160 LET w=60*PI/180
 170 FOR j=1 TO 7
 180  LET w1=j*w
 190  LET x(j)=INT (u+r*COS (w1)
+h)
 200  LET y(j)=INT (v+r*SIN (w1)
+h)
 210 NEXT j
 220 FOR n=1 TO 20
 230  FOR j=1 TO ■
 240   PLOT x(j),y(j)
 250   DRAW x(j+1)-x(j),y(j+1)-y
(j)
 260  NEXT j
 270  FOR k=1 TO 6
 280   LET a(k)=INT ((x(k)+x(k+1
))/2+h)
 290   LET b(k)=INT ((y(k)+y(k+1
))/2+h)
 300  NEXT k
 310  LET a(7)=a(1) : LET b(7)=b
(1)
 320  FOR j=1 TO 7
 330   LET x(j)=a(j) : LET y(j)=
b(j)
 340  NEXT j
 350 NEXT n
 360 IF INKEY$="" THEN GO TO 360
 370 STOP
```



Zie de appendix voor een speciaal effect met dit programma, waarin gebruik gemaakt wordt van een machinetaalprogramma voor het kleuren van vlakken.

Programma 5 tekent alle diagonalen in een n-hoek. De n hoekpunten liggen op de omtrek van een ellips of van een cirkel afhankelijk van de waarden die we in het begin van het programma voor de halve lange as a en halve korte as b kiezen. Voor de berekening van de coördinaten van de punten op de omtrek van de ellips gebruiken we niet de (cartesische) vergelijking $(x^2/a^2) + (y^2/b^2) = 1$, maar de parametervorm $x = a.\cos\varphi$ en $y = b.\sin\varphi$.

Als de parameter φ loopt van 0° tot 360°, dan beschrijft het daaraan toegevoegde punt $P(x,y)$, met $x = a\cos\varphi$ en $y = b\sin\varphi$, precies de omtrek van een ellips. Hieronder zien we een bepaald punt $P(x_1,y_1)$ op de omtrek van de ellips met de daarbij horende waarde van de parameter φ_1. Voor dat punt geldt: $x_1 = a\cos\varphi_1$ en $y_1 = b\sin\varphi_1$.



Als $x_1 = a\cos\varphi_1$ en $y_1 = b\sin\varphi_1$ dan geldt ook

$x_1^2 = a^2\cos^2\varphi_1$ en $y_1^2 = b^2\sin^2\varphi_1$ en ook

$\frac{x_1^2}{a^2} = \cos^2\varphi_1$ en $\frac{y_1^2}{b^2} = \sin^2\varphi_1$, dus dan is

$$\frac{x_1^2}{a^2} + \frac{y_1^2}{b^2} = \cos^2\varphi_1 + \sin^2\varphi_1$$

en omdat $\cos^2\varphi_1 + \sin^2\varphi_1$ gelijk is aan 1 geldt $\frac{x_1^2}{a^2} + \frac{y_1^2}{b^2} = 1$
en hebben we de cartesische relatie tussen x_1 en y_1 afgeleid uit de parametervorm.

Voor het berekenen van de coördinaten van het j-de hoekpunt (x(j),y(j)) verdelen we de maximale middelpuntshoek van 360° in n gelijke hoeken van elk 360/n graden.

Als a de lengte van de halve lange as is èn
b de lengte van de halve korte as èn
n het aantal hoekpunten van de n-hoek èn
w gelijk is aan 360/n èn
w1 de hoek die hoort bij het j-de hoekpunt èn
x(j) de x-coördinaat van het j-de hoekpunt t.o.v het middelpunt van de ellips èn
y(j) de y-coördinaat van het j-de hoekpunt t.o.v. het middelpunt van de ellips

dan zou je in een BASIC-programma de coördinaten x(j) en y(j) als volgt kunnen berekenen:

```
w    = 360/n  :  w1 = (j-1)*w
x(j) = INT(a*COS(w1))
y(j) = INT(b*SIN(w1))
```

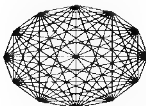
Als we dit doen maken we twee fouten. De eerste fout is dat de hoek w1, als argument van de COSinus- en de SINusfunctie, in graden is uitgedrukt, terwijl BASIC vereist dat het argument van deze functies in radialen wordt uitgedrukt. De tweede fout is dat het grafische coördinatenstelsel van de ZX Spectrum de oorsprong van het coördinatenstelsel voor het scherm in de linkerbenedenhoek van het scherm verwacht en niet in het middelpunt van de ellips. Wat we dus nog moeten doen is:
a. bereken w1 in radialen èn
b. transformeer x(j) en y(j) naar beeldschermcoördinaten.

We weten dat een hoek van 360° overeenkomt met 2π radialen, waarin $\pi = 3{,}14159.....$. Dit betekent dat een hoek van 1 graad overeenkomt met een hoek van $2\pi/360$ of $\pi/180$ radialen. De transformatieformules voor de transformatie van 'onze' coördinaten naar beeldschermcoördinaten hebben we op p.4 gegeven. De drie BASIC-opdrachten voor het berekenen van de beeldschermcoördinaten van het j-de hoekpunt van de n-hoek worden nu:

```
w = (360/n)* PI/180 : w1 = (j-1)*w
x(j) = INT(u + a*COS(w1) + h)
y(j) = INT(v + b*SIN(w1) + h)
```

We gebruiken de BASIC-functie PI.

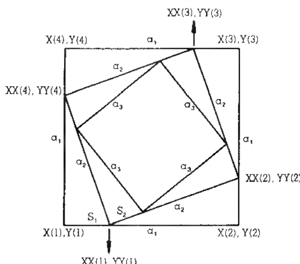
In het navolgende (en ook in het voorgaande) programma komen we bovenstaande opdrachten tegen. Kiezen we voor a en b dezelfde waarde dan krijgen we een regelmatige n-hoek met al zijn diagonalen. Bovendien is de ellips dan een cirkel geworden.



```
100 REM programma 5
     diagonalen in een n-hoek
110 CLS
120 INPUT "HALVE GROTE AS A<=12
7  ";a
130 INPUT "HALVE KLEINE AS B<=8
7  ";b
140 INPUT "HOEVEEL HOEKPUNTEN N
<25  ";n
150 CLS
160 DIM x(n) : DIM y(n)
170 LET u=128 : LET v=88
180 LET h=0.5
190 LET w=(360/n)*PI/180
200 FOR j=1 TO n
210  LET w1=(j-1)*w
220  LET x(j)=INT (u+a*COS (w1)
+h)
230  LET y(j)=INT (v+b*SIN (w1)
+h)
240 NEXT j
250 FOR i=1 TO n-1
260  FOR j=i+1 TO n
270   PLOT x(i),y(i)
280   DRAW x(j)-x(i),y(j)-y(i)
290  NEXT j
300 NEXT i
310 IF INKEY$="" THEN GO TO 310
320 STOP
```

Programma 6 is wat veeleisender in die zin dat het wat voorbereiding nodig heeft.

We willen een reeks ingeschreven vierkanten zo tekenen dat de hoekpunten van een ingeschreven vierkant uit de reeks op de zijden liggen van zijn voorganger. We zien deze situatie voor het eerste, tweede en derde vierkant van de reeks in de onderstaande tekening. De afstand tussen een hoekpunt van een vierkant en een hoekpunt van het volgende vierkant (S_1 en S_2) is steeds een vast deel van de zijde waarop de hoekpunten liggen. Zo is $S_1 = a_1/k$ en $S_2 = a_2/k$. In het algemeen is $S_n = a_n/k$, waarbij a_n de zijde is van het n-de vierkant in de reeks en k de verkleiningsfactor. De waarde voor k kunnen we zelf kiezen. k=16 geeft een fraaie tekening.



(X(1),Y(1)); (X(2),Y(2)); (X(3),Y(3)) en (X(4),Y(4)) zijn de coördinaten van de hoekpunten van een bepaald vierkant uit de reeks. Hoe berekenen we nu de coördinaten van de hoekpunten van het volgende ingeschreven vierkant ((XX(1),YY(1)); (XX(2),...)?

Voor het eerste hoekpunt van het eerste ingeschreven vierkant geldt:

$$XX(1) = X(1) + \frac{X(2)-X(1)}{K} \quad \text{èn}$$

$$YY(1) = Y(1) + \frac{Y(2)-Y(1)}{K}$$

Ga na dat dit klopt in bovenstaande tekening.

Voor het tweede hoekpunt van het ingeschreven vierkant geldt:

$$XX(2) = X(2) + \frac{X(3)-X(2)}{K} \quad \text{èn}$$

$$YY(2) = Y(2) + \frac{Y(3)-Y(2)}{K}$$

In het algemeen geldt:

$$XX(J) = X(J) + \frac{X(J+1)-X(J)}{K} \quad \text{èn}$$

$$YY(J) = Y(J) + \frac{Y(J+1)-Y(J)}{K}$$

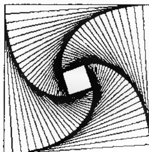
voor J = 1,2,3,4, waarbij X(5) = X(1) en
Y(5) = Y(1).

Als we het tweede vierkant getekend hebben, veranderen we XX(1) in X(1), YY(1) in Y(1), XX(2) in X(2), YY(2) in Y(2), enzovoorts, en tenslotte X(5) in X(1) en Y(5) in Y(1) en we gaan met dezelfde formules opnieuw XX(1),YY(1),... enz. berekenen, maar dan voor de hoekpunten van het volgende vierkant. We zien dit gebeuren in de regels 290 t/m 340 van het volgende programma. De regels 210 t/m 240 tekenen het vierkant, terwijl de regels 250 t/m 280 de hoekpunten van het volgende vierkant berekenen. Een structuurdiagram voor het programma ziet er zo uit:

begin programma ingeschreven vierkanten	
scherm schoonmaken	
lees waarde voor K in	
bepaal coördinaten voor het eerste vierkant	
voor het eerste t/m 40-ste vierkant doe	
	teken dit vierkant
	bepaal coördinaten voor het volgende vierkant
einde programma	

Hier komt het programma:

```
 100 REM programma 8
       ingeschreven vierkanten
 110 CLS
 120 INPUT "TOETS K IN 1<K<20   "
;k
 130 LET h=0.5
 140 DIM x(5) : DIM y(5)
 150 DIM a(5) : DIM b(5)
 160 FOR j=1 TO 5
 170  READ x(j),y(j)
 180 NEXT j
 190 CLS
 200 FOR n=1 TO 40
 210  FOR j=1 TO 4
 220   PLOT x(j),175-y(j)
 230   DRAW x(j+1)-x(j),y(j)-y(j
+1)
 240  NEXT j
 250  FOR j=1 TO 4
 260   LET a(j)=x(j)+INT ((x(j+1
)-x(j))/k+h)
 270   LET b(j)=y(j)+INT ((y(j+1
)-y(j))/k+h)
 280  NEXT j
 290  FOR j=1 TO 4
 300   LET x(j)=a(j)
 310   LET y(j)=b(j)
 320  NEXT j
 330  LET x(5)=x(1)
 340  LET y(5)=y(1)
 350 NEXT n
 360 IF INKEY$="" THEN GO TO 360
 370 STOP
 380 DATA 40,175,215,175,215,0
 390 DATA 40,0,40,175
```



U kunt dit programma als uitgangspunt voor een uitgebreider programma gebruiken. Verdeel het beeldscherm in een aantal vierkanten. In elk van deze vierkanten tekent u, met bovenstaand programma, een reeks van ingeschreven vierkanten. Teken steeds identieke reeksen, of kies steeds een andere waarde voor k, of probeer ze ten opzichte van elkaar te laten draaien.

2 *Grafieken van functies in cartesische vorm*

In het eerste hoofdstuk hebben we alleen rechtlijnige patronen getekend, bijvoorbeeld het programma voor het tekenen van alle diagonalen in een regelmatige n-hoek.

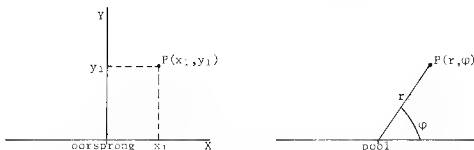
In dit en het volgende hoofdstuk zullen we ons uitvoerig bezig houden met het tekenen van de grafiek van een aantal continue en niet-continue functies. In de wiskunde noemen we de grafiek van een niet-lineaire functie een kromme. De grafiek van een lineaire functie (bijvoorbeeld de functie $y = 4x + 3$) noemen we een rechte. We kunnen de vergelijking van zo'n kromme op drie manieren formuleren:

A ; met cartesische coördinaten	: $y = f(x)$
B ; met poolcoördinaten	: $r = f(\varphi)$
C ; of in parametervorm	: $x = f(t)$, $y = g(t)$

Niet-wiskundigen kennen vaak alleen de gewone (cartesische) vorm $y = f(x)$. Een voorbeeld van een niet-lineaire functie in cartesische vorm is de functie $y = 2x^2 - 3x + 4$. De grafiek van deze functie is een parabool.

De functie $r = 110.\cos(4\varphi)$ stelt een kromme in poolcoördinaten voor. Als we de hoek φ laten lopen van 1 tot 360 graden en we tekenen bij elke hoek φ onder die hoek een punt op afstand r van de oorsprong, dan krijgen we een bloemfiguur als op p.33 staat. Deze vorm $r = 110.\cos(4\varphi)$ heet de poolcoördinatenvorm. Een punt in het platte vlak wordt nu niet gekenmerkt door de afstand van dat punt tot de x- en y-as (cartesisch), maar door de afstand tot de oorsprong (r) en de hoek φ die de x-as maakt met de lijn die dat punt met de oorsprong verbindt. Poolcoördinaten komen in hoofdstuk 3 aan de orde (zie tekening op p.16).

Er is nog een manier om de vergelijking van een kromme weer te geven en dat is de parametervorm. Hierbij worden de x- en y-coördinaat van een punt op de kromme afhankelijk gemaakt van een



Hetzelfde punt P in een
cartesisch coördinatenstelsel en een poolcoördinatenstelsel

bepaalde parameter. Een voorbeeld zijn de vergelijkingen

$x = a.\cos(t)$ en $y = a.\sin(t)$.

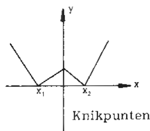
Als we t laten lopen van 0 tot 360°, beschrijven de daarbij horende punten (x,y) precies de omtrek van een cirkel met straal a. De cartesische vorm van deze cirkelvergelijking is $x^2 + y^2 = a^2$, die velen direct zullen herkennen als de 'cirkelvergelijking'.

In dit hoofdstuk houden we ons dus alleen bezig met het tekenen van grafieken van functies en relaties die door middel van cartesische coördinaten beschreven worden. De andere twee vormen komen in hoofdstuk 3 aan de orde.

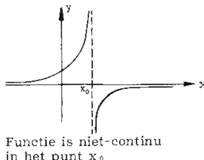
Continue functies

Een continue functie is een functie waarvan de grafiek in één vloeiende beweging van ons 'potlood', dat wil zeggen zonder het potlood van het papier te hoeven halen, getekend kan worden. Er mogen 'knikken' in voorkomen maar geen onderbrekingen (zie tekening op p.17).

De linkergrafiek kunnen we in één beweging, zonder het potlood van het papier te halen, tekenen; bij de rechter grafiek kan dat niet. De linker grafiek is de grafiek van een continue functie; de rechter van een niet-continue functie. Als we alleen links of alleen rechts van het punt x_0 kijken dan is de functie op die intervallen natuurlijk wel continu! De functie is alleen niet-continu in het punt x_0.



Knikpunten



Functie is niet-continu in het punt x_0

We willen nu een algemeen programma ontwikkelen dat, gegeven de grenzen a en b van een bepaald interval ($a \leq x \leq b$) de grafiek tekent van een willekeurige continue functie y = f(x). Mochten de x-as en de y-as in het gebied liggen waarin de grafiek van de functie wordt getekend, dan willen we dat ook beide assen door het programma getekend worden. Om het programma kort, maar toch algemeen, te houden brengen we de volgende vereenvoudigingen aan.

1. De vergelijkingen van de functie, die getekend moet worden, wordt in een subroutine, vanaf regel 1000, opgenomen. De functiewaarde y = f(x), bij een bepaalde x-waarde, wordt berekend door op dat moment met de gewenste x-waarde de subroutine aan te roepen (GO SUB 1000). Er wordt dus niet gebruik gemaakt van de methode waarin de functie als tekst (INPUT-opdracht met stringvariabele) wordt ingelezen waarna het programma deze tekst zelf omzet in een goede BASIC functiedefinitie. Een andere mogelijkheid zou zijn om de functie met een DEF FN-opdracht in het programma te definiëren.

2. Op de coördinaatassen wordt niet automatisch een schaalverdeling aangebracht en ook wordt er geen tekst bij afgedrukt. Veel micro's met HRG*-mogelijkheden kunnen niet tegelijkertijd tekenen en 'schrijven'. Wilt u toch graag weten wat bij een bepaalde waarde van x de functiewaarde (y) is, dan kunt u bijvoorbeeld na het tekenen van de grafiek de computer (op de printer) een lijstje met x- en y-waarden laten afdrukken.

Terug naar het programma-ontwerp. In het eerste deel van het programma berekent de computer, na het inlezen van de waarden van de intervalgrenzen a en b, de grootste en de kleinste functiewaarde (y-waarde) in het opgegeven interval [a,b]. De waarde van hp

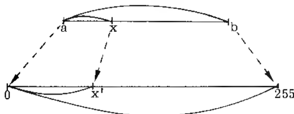
*HRG = High Resolution Graphics

(hoogste punt) en lp (laagste punt) worden op het beeldscherm afgedrukt. We weten zo in welk gebied de computer gaat tekenen. Hierna vraagt het programma of we hp nog groter en of we lp nog kleiner willen maken om daarmee een fraaiere tekening te krijgen. Zo niet, dan toetsen we voor hp en lp dezelfde waarden in als het programma heeft berekend; anders toetsen we de door ons gewenste maximale en minimale functiewaarden in.

Het volgende programmadeel (regels 290-380 op p.20) bevat de lus (FOR x=a TO b STEP dx) voor het tekenen van de grafiek. De coördinaten x,y van een punt op de grafiek moeten met de juiste transformatieformules omgezet worden in beeldschermcoördinaten.

Om het interval [a,b] voor te kiezen x-waarden ($a \leq x \leq b$) om te zetten in het HRG-interval [0,255] ($0 \leq x' \leq 255$) gebruiken we de volgende evenredigheid:

$$(x-a) : (b-a) = x' : 255 \quad \text{ofwel} \quad \frac{x-a}{b-a} = \frac{x'}{255}$$



in BASIC:

```
xx = INT(kx*(x-a)+h),    met kx = 255/(b-a).
```

We doen dit ook voor het afbeelden van het functiebereik $lp \leq y \leq hp$ op het grafische beeldbereik $0 \leq y' \leq 175$:

$$(y-lp) : (hp-lp) = y' : 175 \quad \text{ofwel} \quad \frac{y-lp}{hp-lp} = \frac{y'}{175}$$

Dit geeft in BASIC:

```
yy = INT(ky*(y-lp)+h) met ky = 175/(hp-lp).
```

We zien deze berekeningen van xx en yy in de regels 310 en 320. In de volgende regels worden twee, op deze wijze berekende, punten (x1,y1) en (x2,y2) door een recht lijntje met elkaar verbonden.

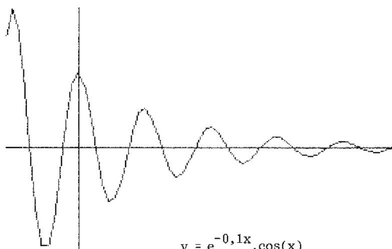
Vervolgens (regels 390-420) wordt de ligging van de x- en y-as bepaald. Kiezen we in bovenstaande transformatievergelijkingen

voor x en y de waarde nul, dan vinden we respectievelijk de ligging van de y-as en van de x-as (regels 390 en 420).

Met deze uitleg is de werking van het programma hopelijk te volgen.

Test u het programma 7 met willekeurige, maar continue, functies. In dit voorbeeldprogramma hebben we de functie $y = e^{-0,1x}.\cos(x)$ gekozen, hetgeen de grafiek van een gedempte trilling oplevert. Hieronder zijn enkele ideeën voor minder moeilijke functies.

functie	regel 1000	waarde voor a	waarde voor b
$y = \sin(x)$	y = SIN(x)	0	2Π (= 6,2832)
$y = x^2$	y = x*x	-4	+4
$y = e^x$	y = EXP(x)	-3	+3
$y = x^3-2x^2-x$	y = x*x*x-2*x*x-x	-1	+3



$y = e^{-0,1x}.\cos(x)$

(zie regel 1000 van programma 7)

```
 100 REM programma 7  grafiek
     van een continue functie
 110 CLS
 120 INPUT "LINKER-INTERVAL GREN
S   ";a
 130 INPUT "RECHTER-INTERVAL GRE
NS   ";b
 140 IF a>b THEN LET c=a: LET a=
b: LET b=c
 150 LET hp=-100000: LET lp=1000
00
 160 LET dx=(b-a)/64
 170 FOR x=a TO b STEP dx
 180  GO SUB 480
 190  IF y>hp THEN LET hp=y
 200  IF y<lp THEN LET lp=y
 210 NEXT x
 220 PRINT "GROOTSTE Y-WAARDE: "
;hp
 230 PRINT "KLEINSTE Y-WAARDE: "
;lp
 240 INPUT "BOVENGRENS VOOR Y  "
;hp
 250 INPUT "ONDERGRENS VOOR Y  "
;lp
 260 CLS
 270 LET kx=255/(b-a): LET ky=17
5/(hp-lp)
 280 LET h=0.5
 290 FOR x=a TO b STEP dx
 300  GO SUB 1000
 310  LET xx=INT (kx*(x-a)+h)
 320  LET yy=INT (ky*(y-lp)+h)
 330  IF x=a THEN LET x1=xx: LET
y1=yy: GO TO 380
 340  LET x2=xx: LET y2=yy
 350  PLOT x1,y1
 360  DRAW x2-x1,y2-y1
 370  LET x1=x2: LET y1=y2
 380 NEXT x
 390 LET x1=0: LET y1=INT (ky*(-
lp)+h)
 400 LET x2=255: LET y2=y1
 410 IF y1>0 AND y1<255 THEN PLO
T x1,y1: DRAW x2-x1,y2-y1
 420 LET x1=INT (kx*(-a)+h): LET
y1=0
 430 LET x2=x1: LET y2=175
 440 IF x1>=0 AND x1<=255 THEN P
LOT x1,y1: DRAW x2-x1,y2-y1
 450 IF INKEY$="" THEN GO TO 450
 460 STOP
 470
1000 LET y=EXP (-0.1*x)*COS (x)
1010 RETURN
```

Nu volgen drie korte programma's.

Programma 8 is de eerste van deze drie. Dit programma tekent tien in fase verschoven sinuskrommen, allemaal in hetzelfde coördinatenstelsel. De vergelijking van het stelsel is

$$y = \sin(x+np), \quad \text{met als fase } p = \frac{\pi}{9}\ (20°).$$

We vinden de tien vergelijkingen door voor n de waarden 0,1,2 t/m 9 te kiezen. De lusvariabele j (regel 150) komt overeen met de beeldschermcoördinaat xx. Omdat we in dit programma de beeldschermcoördinaat xx, dat wil zeggen j, in stappen van 5 naar 255 laten lopen (regel 150), moeten we eerst de bij j behorende waarde voor x berekenen. Dan pas kunnen we de functiewaarde y = f(x) berekenen. We kunnen de evenredigheid

$$(x-a) : (b-a) = j : 255$$

natuurlijk ook gebruiken om x uit j te berekenen; hieruit volgt:

$$x = j.\frac{(b-a)}{255} + a.$$

In het onderstaande programma zijn de gekozen intervalgrenzen $a = 0$ en $b = 2\pi$, zodat bovenstaande vergelijking wordt:

$$x = j.\frac{2\pi}{255}.$$

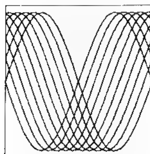
Als we $c = \frac{2\pi}{255}$ nemen, wordt in het programma het argument van de sinusfunctie dus j*c+n*p.

In het programma wordt de waarde j*c als x berekend, dus zien we in regel 170 x+n*p als argument van de SINusfunctie. Natuurlijk moeten we de functiewaarde y = SIN(x+n*p) nog omzetten naar de beeldschermcoördinaat y, zodat we krijgen

y = INT(v+k*SIN(x+n*p)+h).

Door voor de schaalconstante k de waarde 87 te kiezen (regel 120) krijgen we een mooie 'grote' tekening.

Wie met kleuren wil werken kan dit programma uitbreiden door de diverse krommen andere kleuren te geven. Iets moeilijker zal het zijn de 'banen' tussen de sinuskrommen in te kleuren. Probeer het eens.



```
 100 REM programma 6
          10 sinuskrommen
 110 CLS
 120 LET v=87: LET k=87: LET h=0
.5: LET p=PI/9
 130 LET c=2*PI/255
 140 FOR n=0 TO 9
 150   FOR j=0 TO 255 STEP 5
 160     LET x=j*c
 170     LET y=INT (v-k*SIN (x+n*p
)+h)
 180     IF j=0 THEN LET x1=j: LET
 y1=y: GO TO 230
 190     LET x2=j: LET y2=y
 200     PLOT x1,175-y1
 210     DRAW x2-x1,y1-y2
 220     LET x1=x2: LET y1=y2
 230   NEXT j
 240 NEXT ■
 250 IF INKEY$="" THEN GO TO 250
 260 STOP
```

De illustraties zoals hierboven en op de pagina hiernaast zijn door de auteur gemaakt op een plotter, waarbij de linkerbovenhoek de oorsprong is en niet zoals bij de ZX Spectrum de linkerbenedenhoek. Om hetzelfde effect te krijgen als de illustraties aangeven moeten we in sommige programma's ook uitgaan van een oorsprong linksboven. Kijk welke figuur u krijgt als u in regel 170 niet v-k*SIN maar v+k*SIN neemt en in regel 200 niet PLOT x1,175-y1 maar PLOT x1,y1 en in regel 210 niet DRAW x2-x1,y1-y2 maar DRAW x2-x1,y2-y1.

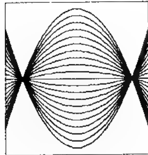
ZX Spectrum kleurcodes

Code	Kleur	Code	Kleur
0	zwart	4	groen
1	blauw	5	vaalblauw
2	rood	6	geel
3	paars	7	wit

Programma 9 tekent een stelsel parabolen met de vergelijking

$y = -tx^2 + t$

Alle parabolen (bepaalde t-waarden) hebben dezelfde snijpunten met de x-as (x = 1 en x = -1).



```
 100 REM programma 9
          paraboolstelsel
 110 CLS
 120 LET u=128: LET v=87: LET h=
0.5
 130 FOR k=-87 TO 87 STEP 10
 140   FOR x=-110 TO 110 STEP [illegible]
 150     LET xx=INT (u+x+h)
 160     LET y=-k*x*x/6400+k: LET
y=INT (v+y+h)
 170     IF x=-110 THEN LET x1=xx:
 LET y1=y: GO TO 210
 180     LET x2=xx: LET y2=y
 190     PLOT x1,y1: DRAW x2-x1,y2
-y1
 200     LET x1=x2: LET y1=y2
 210   NEXT x
 220 NEXT k
 230 IF INKEY$="" THEN GO TO 230
 240 STOP
```

Voeg toe 145 IF x <= 0 THEN INK 1 : GOTO 150
146 INK 6

of 145 IF INT((k/10)/2)*2=k/10 THEN INK 1 : GOTO 150
146 INK 6

om wat kleur in de tekening aan te brengen.

In dit programma maakt het niet uit of de oorsprong linksboven of linksonder ligt. Het plaatje is symmetrisch rond de horizontale lijn midden in het scherm. We kunnen in regel 160 y=INT(v-y+h) nemen en in regel 190 PLOT x1,175-y1: DRAW x2-x1,y1-y2; dit geeft dezelfde figuur, die alleen van bovenaf getekend wordt in plaats van onderaf.

Soms willen we de oppervlakte tussen de grafiek van een functie en de x-as berekenen (de integraal bepalen) of laten tekenen.

Programma 10 kleurt zo'n oppervlak tussen de x-as en de grafiek van de functie

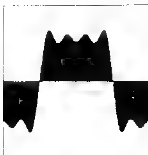
$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}\ .$$

Ook nu is de lusvariabele j de grafische coördinaat xx. De waarde voor a is $-\pi$ en die voor b is $+\pi$, waaruit volgt dat (zie p.21)

$$x = \frac{j.2\pi}{255} - \pi\ .$$

Ook nu nemen we in het programma $c = \frac{2\pi}{255}$

```
 100 REM programma 10
     oppervlakte onder een kromme
 110 CLS
 120 LET v=88: LET h=0.5
 130 LET k=80: LET c=2*PI/255
 140 FOR j=0 TO 255
 150  LET x=j*c-PI: GO SUB 1000
 160  LET y=INT (v+k*y+h)
 170  PLOT j,v
 180  DRAW 0,y-v
 190 NEXT j
 200 IF INKEY$="" THEN GO TO 200
 210 STOP
 220
1000 LET y=COS (x)-COS (3*x)/3+C
OS (5*x)/5-COS (7*x)/7
1010 RETURN
```



Probeer de regels 160, 170 en 180 zo te veranderen, dat u een figuur krijgt die gespiegeld is ten opzichte van de horizontale lijn in het midden van het scherm (kijk naar programma 8).

Probeer dit programma zo te maken dat de drie oppervlakken in bijvoorbeeld rood, wit en blauw gekleurd worden.

Niet-overal-continue functies

De functie $y = \frac{x^2+3}{x^2-x-6}$ kan niet met behulp van programma 7 getekend worden. Zouden we bijvoorbeeld voor x het interval $-5 \leq x \leq 5$ kiezen, dan liggen hierin de punten x = 3 en x = -2. Dit zijn de twee punten waarvoor de noemer van bovenstaande functie nul is. Als de computer bij het tekenen bij één van deze twee punten belandt, zal op het scherm een foutmelding (DIVISION BY ZERO) 'delen door nul' verschijnen en zal het programma afgebroken worden.

Dit zal ook gebeuren bij logaritmische functies met een niet-positief argument of bij wortelfuncties met een negatief argument. Zelfs bij het tekenen van continue (nette) functies kunnen problemen optreden. We komen hierbij in de problemen als de functiewaarden heel groot of heel klein worden. Transformatie van dergelijke waarden naar beeldschermcoördinaten (0-255 en 0-175) heeft dan geen enkele zin meer, omdat de aard van het verloop van de functie in het geheel niet meer tot uitdrukking komt. Het is daarom beter om de functiewaarden van een continue functie naar boven en beneden te begrenzen. Dit heeft tot gevolg dat de grafiek van een continue functie er uit zou kunnen zien als de grafiek van een niet-continue functie.

Het zal duidelijk zijn dat je een algemeen programma voor het tekenen van een willekeurige continue of niet-continue functie niet zomaar even opschrijft. Dergelijke programma's kom je in de vakliteratuur dan ook niet of nauwelijks tegen, en mocht je er wel een tegenkomen, dan betreft het òf een heel groot programma òf een programma dat voor een bepaalde functie, waarvan men van te voren weet waar de discontinuïteiten zitten, geschreven is.

We geven nu een verbazend kort programma voor het tekenen van de grafiek van een willekeurige functie y = f(x). Veel wiskundeleraren, scholieren en studenten zullen nu hun 'oren' spitsen. U begrijpt dat, gezien het bovenstaande, hierbij wel enkele beperkingen gelden:

1. Degene die het programma gaat gebruiken moet een beetje kunnen programmeren. De functie moet namelijk in gedeelten in een subroutine (vanaf regel 1000) beschreven worden.
2. De programmagebruiker moet voldoende wiskundige kennis bezitten om te kunnen bepalen voor welke x-waarden functies moeilijkheden kunnen opleveren.

We zullen zien dat, normaal gesproken, slechts een paar BASIC-regels nodig zijn om de functiebeschrijving te programmeren. Het berekenen van nulpunten met behulp van een of ander numeriek-wiskundig algoritme is in het geheel niet nodig.

In het onderstaande **programma 11** gebruiken we twee variabelen fz en fa als zogeheten vlaggen (Flags). Een vlag is een variabele die slechts twee waarden (vaak 0 en 1) kan aannemen.

Om de werking van de vlak fz duidelijk te maken geven we hieronder de subroutine 1000 met de functiebeschrijving van de functie

$$y = \frac{x^2+3}{x^2-x-6}$$

```
1000 LET n=x*x-x-6
1010 IF n=0 THEN LET fz=1: RETURN
1020 LET y=(x*x+3)/n
1030 IF y<lp OR y>hp THEN LET fz=1: RETURN
1040 LET fz=0: RETURN
```

De vlak fz wordt alleen op 1 gezet als het punt (x,y) niet op het scherm getekend kan worden. Dit is het geval als de functie niet-continu is in het punt (x,y) ($n = 0$; $x = 3$ en $x = -2$) of als de functiewaarde buiten het opgegeven functiebereik ($lp \leq y \leq hp$) valt.

Waarvoor dient nu de tweede vlag fa? Bekijk het volgende stukje programma eens:

```
200 LET fa=1
210 FOR x=a TO b STEP dx
220   LET x2=INT(kx*(x-a)+h): GO SUB 1000
230   IF fz=1 THEN LET fa=1: GO TO 310
240   IF fa=1 THEN GO TO 290
250   LET y2=INT(ky*(y-lp)+h)
260   PLOT x1,y1
270   DRAW x2-x1,y2-y1
280   LET x1=x2: LET y1=y2: GO TO 310
290   LET x1=x2: LET y1=INT(ky*(y-lp)+h)
300   LET fa=0
310 NEXT x
```

Als fz=1 dan wordt fa ook gelijk aan 1 gemaakt en wordt de volgende x-waarde bekeken (FOR-lus) zonder dat het nieuw berekende punt met het laatstgetekende punt verbonden wordt. Is fz echter 0 (berekende punt ligt in het beeldvlak) dan wordt onderzocht of fa daarvoor soms op 1 gezet is. Is dit zo (tweede THEN) dan moet het nieuw berekende punt als nieuw beginpunt (x1,y1) gekozen worden en dit mag dan ook niet met het vorige getekende punt verbonden worden; fa wordt in dit geval nul gemaakt.

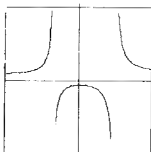
Dus als zowel fz als fa nul zijn, wordt het nieuw berekende punt verbonden met het laatstgetekende punt.

Om te zorgen dat het programma probleemloos werkt moet voor het berekenen van het eerste punt van de grafiek, dus voor de FOR-lus, de vlag fa op 1 gezet worden, waardoor we er zeker van zijn dat de waarden x1 en y1 berekend worden. Hier volgt het volledige programma:

```
 100 REM programma 11 grafiek
    van een willekeurige functie
 110 CLS
 120 INPUT "LINKERGRENS VOOR X
";a
 130 INPUT "RECHTERGRENS VOOR X
";b
 140 INPUT "BOVENGRENS VOOR Y  "
;hp
 150 INPUT "ONDERGRENS VOOR Y  "
;lp
 160 IF a>b THEN LET c=a: LET a=
b: LET b=c
 170 LET kx=255/(b-a): LET ky=17
5/(hp-lp): LET h=0.5
 180 LET dx=(b-a)/256
 190 CLS
 200 LET fa=1
 210 FOR x=a TO b STEP dx
 220  LET x2=INT (kx*(x-a)+h): G
O SUB 1000
 230  IF fz=1 THEN LET fa=1: GO
TO 310
 240  IF fa=1 THEN GO TO 290
 250  LET y2=INT (ky*(y-lp)+h)
 260  PLOT x1,y1
 270  DRAW x2-x1,y2-y1
 280  LET x1=x2: LET y1=y2: GO T
O 310
 290  LET x1=x2: LET y1=INT (ky*
(y-lp)+h)
 300  LET fa=0
 310 NEXT x
 320 LET x1=0: LET y1=INT (ky*(-
lp)+h)
 330 LET x2=255: LET y2=y1
 340 IF y1>0 AND y1<=175 THEN PL
OT x1,y1: DRAW x2-x1,y2-y1
 350 LET x1=INT (kx*(-a)+h): LET
 y1=0
 360 LET x2=x1: LET y2=175
 370 IF x1>0 AND x1<=255 THEN PL
OT x1,y1: DRAW x2-x1,y2-y1
 380 IF INKEY$="" THEN GO TO 380
 390 STOP
 400
1000 LET n=x*x-x-6
1010 IF n=0 THEN LET fz=1: RETUR
N
1020 LET y=(x*x+3)/n
1030 IF y<lp OR y>hp THEN LET fz
=1: RETURN
1040 LET fz=0: RETURN
```

Dit programma tekent de grafiek van de functie

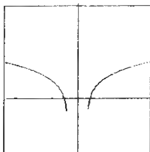
$y = \frac{x2+3}{x^2-x-6}$; kies voor a, b, hp en lp resp.
-5, 5, 10 en -10



Wilt u een andere functie tekenen, bijvoorbeeld $y = \ln(x^2-2)$, herschrijf dan subroutine 1000 (regels 1000,,1010 en 1020):

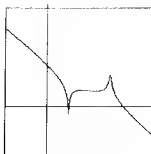
```
1000 LET u=x*x-2: IF u<=0 THEN LET fz=1: RETURN
1010 LET y=LOG(u)
```

U krijgt dan deze grafiek:



Kies voor a, b, hp en lp de waarden -25, 25, 8 en -5.

Hier volgt een hele fraaie:



$y = 3 - x + \ln\left|\frac{x-1}{x-3}\right|$

Verander de regels 1000 en 1010 in:

```
1000 LET n=x-3: IF n=0 THEN LET fz=1: RETURN
1010 LET y=3-x+LOG(ABS((x-1)/n))
```

Kies voor a, b, hp en lp de waarden -5, 10, 8 en -4.

3 *Krommen in poolcoördinaten en in parametervorm*

In het vorige hoofdstuk hebben we ons beziggehouden met het tekenen van de grafiek van een continue of niet-continue functie, waarvan de vergelijking als $y = f(x)$ geschreven kon worden; dus in cartesische vorm. Nu gaan we grafieken tekenen van functies waarvan de vergelijking in poolcoördinaten geschreven wordt of in de zogeheten parametervorm. Dit levert zeer fraaie 'beelden' op.

Krommen met poolcoördinaten

Functies, waarvan de grafiek een gesloten kromme laat zien, zijn vaak eenvoudiger met poolcoördinaten te beschrijven dan in cartesische vorm. Als voorbeeld noemen we de 'hartkromme' (kardioide), waarvan de vergelijking in poolcoördinaten eenvoudig is, namelijk:

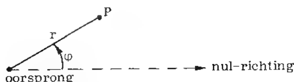
$$r = k(1 + \cos\varphi),$$

maar waarvan de cartesische vorm nogal ingewikkeld is, namelijk

$$\frac{(x^2-y^2-kx)^2}{k^2(x^2+y^2)} = 1.$$

Om de volgende programma's te doorgronden (niet om ze te draaien!) is een beetje theorie nodig. Het poolcoördinatenstelsel wordt in de wiskundelessen op school niet of nauwelijks behandeld. Eigenlijk is dit jammer, want hiermee (en met de parametervorm) kunnen juist de mooiste functies (en daarmee de fraaiste grafieken) beschreven en getekend worden.

In een poolcoördinatenstelsel wordt elk punt in het platte vlak met twee coördinaten, te weten r en φ, bepaald. r is de afstand tussen het punt en de oorsprong en φ (phi, spreek uit: fie) is de hoek tussen de horizontale lijn door de oorsprong en de lijn door de oorsprong en het punt (zie tekening op p. 31).

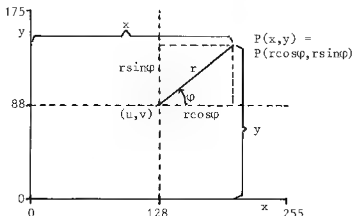


De oorsprong van het poolcoördinatenstelsel leggen we in het middelpunt van het beeldscherm (u=128, v=88). Zoals we weten ligt de oorsprong van het beeldschermcoördinatenstelsel in de linkerbenedenhoek van het beeldscherm (HOME-positie). De nul-richting in ons poolcoördinatenstelsel is horizontaal en wijst naar rechts. φ=90° (π/2 radialen) is verticaal naar boven; φ=180° (π radialen) is horizontaal naar links en φ=270° (3π radialen) verticaal naar beneden.

We gebruiken de volgende twee transformatieformules om het, uit de functievergelijking berekende, punt P(r,φ) om te zetten in het beeldschermpunt P(x,y):

```
x = INT(u + r*COS(p) + 0.5)     èn
y = INT(v + r*SIN (p) + 0.5)
```

In onderstaande figuur zien we hiervoor de verklaring.

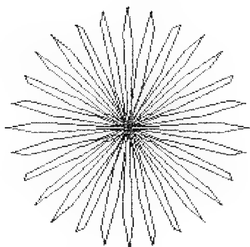


De hoek φ, in de programma's voorgesteld door de variabele p, doorloopt steeds in positieve richting (tegen de klok in) het interval van 0 tot 2π. Dit is een interval in radialen (0-360 in graden).

In alle programma's is als lusvariabele de hoek w in graden gekozen. Met de formule p = w.(π/180) (in de programma's p=w*rd) wordt de hoek in radialen berekend. Het voordeel van een lusvariabele

die het interval 0-360° doorloopt is dat de stapgrootte waarmee de lusvariabele steeds wordt opgehoogd een geheel getal kan zijn (1 bijvoorbeeld) en dat hierdoor de programma's beter leesbaar zijn. Een neveneffect is dat alle programma's bijna woordelijk in Pascal zijn over te zetten; Pascal kent immers geen gebroken stapgrootte in een lus.

Meer theorie hebben we niet nodig. De volgende programma's spreken grotendeels voor zichzelf.

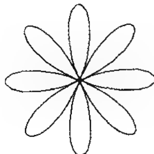


Deze figuur ontstaat als we in programma 12 in regel 1000 LET r=COS(14*p) nemen.

Programma 12 tekent de grafiek van de functie

$$r = k\cos(n\varphi)$$

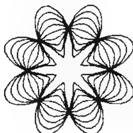
In het programma kiezen we k = 87 en n = 4.



```
 100 REM programma 12 grafiek
    van de functie r=COS (4*phi)
 110 CLS
 120 LET u=128: LET v=88
 130 LET h=0.5: LET k=87
 140 LET rd=PI/180
 150 FOR w=0 TO 360 STEP 3
 160  LET p=w*rd: GO SUB 1000
 170  LET x=INT (u+k*r*COS (p)+h
)
 180  LET y=INT (v+k*r*SIN (p)+h
)
 190  IF p=0 THEN LET x1=x: LET
y1=y: GO TO 230
 200  LET x2=x: LET y2=y
 210  PLOT x1,y1: DRAW x2-x1,y2-
y1
 220  LET x1=x2: LET y1=y2
 230 NEXT w
 240 IF INKEY$="" THEN GO TO 240
 250 STOP
 260
1000 LET r=COS (4*p): LET r=ABS
(r)
1010 RETURN
```

Het programma tekent een achtdelige draaisymmetrische figuur; een bloem met acht blaadjes. Als u met dit programma gaat experimenteren zult u snel ontdekken dat voor even waarden van n een 2n-bladerige en voor oneven waarden van n een n-bladerige bloem ontstaat. Kunt u een tweekleurige bloem maken?

Programma 14 tekent sinuskrommen die cirkelvormig gekromd zijn.



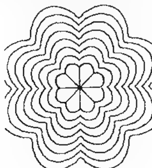
```
 100 REM programma 14
         sinuskrommen in cirkels
 110 CLS
 120 LET u=128: LET v=88
 130 LET h=0.5: LET rd=PI/180
 140 FOR k=-33 TO 33 STEP 8
 150  FOR w=0 TO 360 STEP 2
 160   LET p=w*rd: GO SUB 1000
 170   LET x=INT (u+r*COS (p)+h)
 190   LET y=INT (v+r*SIN (p)+h)
 200   IF p=0 THEN LET x1=x: LET
 y1=y: GO TO 240
 210   LET x2=x: LET y2=y
 220   PLOT x1,y1: DRAW x2-x1,y2
-y1
 230   LET x1=x2: LET y1=y2
 240  NEXT w
 250 NEXT k
 260 IF INKEY$="" THEN GO TO 260
 270 STOP
 280
1000 LET r=60+k*SIN (4*p)
1020 RETURN
```

Probeert u eens:

```
1000 LET r = 2*TAN(2*p)+k*SIN(2*COS(SIN(6*p)))
```

voor een schitterend spinneweb. De variaties zijn echt onbegrensd!

Programma 15 tekent een bloemvormige figuur met blaadjes die in het midden aan steeltjes vastzitten. In het programma is n = 4 gekozen. U kunt gerust andere waarden voor n proberen. Het effect is hetzelfde als bij programma 12.



```
 100 REM programma 15 bloemen
 110 CLS
 120 LET u=128: LET v=88
 130 LET h=0.5: LET rd=PI/180
 135 LET n=4: LET c=0.25:
     REM eerst de bloemen
 140 FOR k=20 TO 69 STEP 7
 150  FOR w=0 TO 360 STEP 3
 160   LET p=w*rd: GO SUB 1000
 170   LET x=INT (u+r*COS (p)+h)
 190   LET y=INT (v+r*SIN (p)+h)
 200   IF p=0 THEN LET x1=x: LET
y1=y: GO TO 240
 210   LET x2=x: LET y2=y
 220   PLOT x1,y1: DRAW x2-x1,y2
-y1
 230   LET x1=x2: LET y1=y2
 240  NEXT w
 250 NEXT k
 260 LET r=20: LET p1=(180/n)*rd
:    REM dan de stelen
 270 FOR j=1 TO n
 280  LET p=j*p1
 290  LET x1=INT (u+r*COS (p)+h)
 300  LET y1=INT (v+r*SIN (p)+h)
 310  LET x2=INT (u+r*COS (p+PI)
+h)
 320  LET y2=INT (v+r*SIN (p+PI)
+h)
 330  PLOT x1,y1: DRAW x2-x1,y2-
y1
 340 NEXT j
 350 IF INKEY$="" THEN GO TO 350
 360 STOP
 370
1000 LET r=k*(1+c*ABS (SIN (n*p)
))
1020 RETURN
```

Bezit u een kleurenmonitor of een kleurenplotter dan kunt u na elke doorloop van de k-lus de afdrukkleur veranderen. Ook de stelen van de bloem kunnen een eigen kleur krijgen. Tekent u een aantal van dergelijke bloemen naast of onder elkaar dan hebt u een begin gemaakt met 'computer-art'.

Spiralen zijn altijd geliefde figuren. Programma 16 tekent logaritmische spiralen of spiralen van Archimedes. Deze laatste soort spiralen hebben als vergelijking:

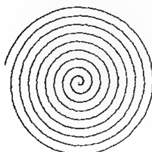
$$r = c.\varphi$$

In het programma op p.40 is voor c de waarde 3 gekozen. Kies gerust andere waarden, maar wel tussen 0,5 en 20.

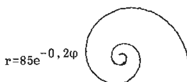
De logaritmische spiraal heeft (naar Bernoulli) de vergelijking

$$r = k\,e^{c\varphi}.$$

In het programma dat de onderste spiraal getekend heeft hebben we voor k de waarde 85 en voor c de waarde -0,2 gekozen.



$r=2\varphi$



$r=85e^{-0,2\varphi}$

Omdat de logaritmische spiraal zich oneindig vaak rond de oorsprong zal winden moet ervoor gezorgd worden dat het programma na bepaalde 'tijd' wordt afgebroken.

```
100 REM programma 16 spiralen
110 CLS
120 LET u=128: LET v=88
130 LET h=0.5: LET rd=PI/180
140 LET c=3
150 FOR w=0 TO 10000 STEP 3
160    LET p=w*rd: GO SUB 1000
170    LET x=INT (u+r*COS (p)+h)
190    LET y=INT (v+r*SIN (p)+h)
200    IF p=0 THEN LET x1=x: LET
y1=y: GO TO 240
205    IF x<0 OR x>255 OR y<0 OR
y>175 THEN GO TO 260
210    LET x2=x: LET y2=y
220    PLOT x1,y1: DRAW x2-x1,y2
-y1
230    LET x1=x2: LET y1=y2
240 NEXT w
260 IF INKEY$="" THEN GO TO 260
270 STOP
280
1000 LET r=c*p
1020 RETURN
```

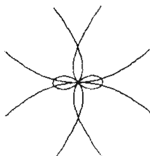
Voeg toe regel 245 u = 132 : GO TO 150 en verander in regel 205 de opdracht GO TO 260 in GO TO 245, en zie hoe dit de figuur verfraait.

Voor het tekenen van een logaritmische spiraal maken we c gelijk aan -0,2 en herschrijven subroutine 1000 als volgt:

```
1000 LET r=85*EXP(c*p)
1010 IF r<5 THEN STOP
1020 RETURN
```

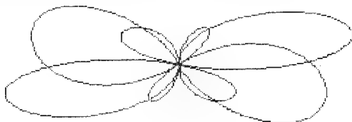
Als laatste programma met poolcoördinaten geven we het programma 17 voor het tekenen van een willekeurige, in poolcoördinaten geformuleerde, continue of niet-continue functie. De programmastructuur komt overeen met de structuur van programma 11 (zie p.27). U kunt daar de werking van de vlaggen fz en fa nog eens bestuderen. Ook de andere variabelen hebben dezelfde betekenis als hun naamgenoten in programma 11. Als voorbeeld in het programma hebben we een vrij ingewikkelde functie, die niet overal continu is, gekozen:

$$r = \frac{\sin(1,5.\varphi)}{1-2.\cos\varphi}.$$



De figuur is getekend met a = -2, b = 2, lp = -2, hp = 2, wo = 0° en wn = 720°. Om de grafiek er optisch mooi uit te laten zien wordt er niet getest op een negatieve waarde van r.

Wilt u een andere functie proberen, verander dan alleen iets in de regels 1000 t/m 1090. Verander de regels 1100 t/m 1200 niet.



Deze zeldzame vlinder heeft als vergelijking:

$$r = \frac{4.\sin(1,5p+2)}{\cos(p).(1+\frac{\cos(3p)}{3})}$$

Voor a, b, lp, hp, wo en wn nemen we in programma 17 de waarden -4, +4, -4, +4, 0 en 720.

```
 100 REM programma 17 grafiek va
n de functie r=f(phi)
 110 CLS
 120 INPUT "LINKERGRENS VOOR X
";a
 130 INPUT "RECHTERGRENS VOOR X
 ";b
 140 INPUT "ONDERGRENS VOOR Y  "
;lp
 150 INPUT "BOVENGRENS VOOR Y  "
;hp
 160 INPUT "STARTWAARDE VOOR PHI
 ";wo
 170 INPUT "EINDWAARDE VOR PHI
";wn
 180 LET kx=255/(b-a): LET ky=17
5/(hp-lp)
 190 LET h=0.5: LET rd=PI/180
 200 CLS
 210 LET fa=1
 220 FOR w=wo TO wn
 230  LET p=w*rd: GO SUB 1000
 240  IF fz=1 THEN LET fa=1: GO
TO 330
 250  IF fa=1 THEN GO TO 300
 260  LET x2=INT (kx*(x-a)+h)
 270  LET y2=INT (ky*(y-lp)+h)
 280  PLOT x1,y1: DRAW x2-x1,y2-
y1
 290  LET x1=x2: LET y1=y2: GO T
O 330
 300  LET x1=INT (kx*(x-a)+h)
 310  LET y1=INT (ky*(y-lp)+h)
 320  LET fa=0
 330 NEXT w
 340 IF INKEY$="" THEN GO TO 340
 350 STOP
1000 LET n=1-2*COS (p): IF n=0 T
HEN LET fz=1: RETURN
1010 LET r=SIN (3*p/2)/n
1100 LET x=r*COS (p): LET y=r*SI
N (p)
1110 IF x<a OR x>b OR y<lp OR y>
hp THEN LET fz=1: RETURN
1200 LET fz=0: RETURN
```

De parametervorm

De meest interessante krommen krijgen we bij functies waarvan de vergelijking in parametervorm wordt opgesteld. Dit houdt in dat zowel de x- als de y-coördinaat als functie van dezelfde, derde, parameter t worden uitgedrukt. In de natuurkunde zien we vaak de tijd als parameter (vandaar de t). In de wiskunde is de parameter t bijna altijd een hoek in radialen.

Zo is de parametervorm van een cirkel met straal r en het middelpunt in de oorsprong:

$$x = r\cos t \qquad \text{en} \qquad y = r\sin t$$

Uit deze twee vergelijkingen kan gemakkelijk de cartesische vorm $x^2+y^2=r^2$ gedistilleerd worden. De vergelijking van een ellips met het middelpunt in de oorsprong en met een halve lange as a en een halve korte as b is:

$$x = a\cos t \qquad \text{en} \qquad y = b\sin t$$

Het is eenvoudig programma's te ontwikkelen voor het tekenen van stelsels concentrische of excentrische cirkels. Ook stelsels ellipsen zijn, dankzij de parametervorm, eenvoudig te tekenen. We doen dit echter niet, omdat de figuren niet zo bijzonder zijn en vrij bekend. In hoofdstuk 1 hebben we trouwens al een ellips met behulp van de parametervorm geprogrammeerd.

De volgende programma's tekenen figuren die u mogelijk nog niet kent en die zich voor computer-graphics bijzonder goed lenen.

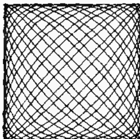
Programma 18 tekent een Lissajousfiguur. In het algemeen is de parametervorm voor een dergelijke figuur:

$$x = k_1.\sin(f_1.t+p_1) + k_2.\cos(f_2.t)$$
$$y = k_3.\sin(f_3.t+p_3) + k_4.\cos(f_4.t)$$

Hierin is p de fase, f de frequentie en k de amplitude. Om een mooie figuur te krijgen hebben we de volgende waarden gekozen:

$k_1=k_3=87$, $f_1=16$, $p_1=0$, $f_3=17$, $p_3=35$ en $k_2=k_4=f_2=f_4=0$

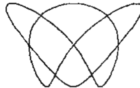
```
 100 REM programma 18
         lissajousfiguren
 110 CLS
 120 PRINT "TOETS K1,F1,P1,K2,F2
IN ";
 130 INPUT k1,f1,p1,k2,f2
 140 PRINT : PRINT
 150 PRINT "TOETS K3,F3,P3,K4,F4
IN ";
 160 INPUT k3,f3,p3,k4,f4
 170 LET u=128: LET v=88
 180 LET h=0.5: LET rd=PI/180
 190 CLS
 200 FOR w=0 TO 360
 210  LET t=w*rd: GO SUB 1000
 220  LET x=INT (u+x+h)
 230  LET y=INT (v+y+h)
 240  IF w=0 THEN LET x1=x: LET
y1=y: GO TO 280
 250  LET x2=x: LET y2=y
 260  PLOT x1,y1: DRAW x2-x1,y2-
y1
 270  LET x1=x2: LET y1=y2
 280 NEXT w
 290 IF INKEY$="" THEN GO TO 290
 300 STOP
 310 :
1000 LET x=k1*SIN (f1*t+p1)+k2*C
OS (f2*t)
1010 LET y=k3*SIN (f3*t+p3)+k4*C
OS (f4*t)
1020 RETURN
```



Experimenteer met het programma! Zoek zelf de juiste waarden voor de constanten zodat u mooie figuren krijgt. Natuurkundigen zullen u hierbij graag helpen; zij weten welke waarden u moet nemen!

Programma 19 presenteert een niet alledaagse figuur. Deze figuur wordt dikwijls de 'vliegenkopfiguur' genoemd. Om de vliegenkop horizontaal op het beeldscherm te krijgen moet de parameter t het interval 90°-450° doorlopen.

```
 100 REM programma 19 vliegenkop
 110 CLS
 120 LET u=128: LET v=87
 130 LET h=0.5: LET k=30: LET rd
=PI/180
 140 FOR w=90 TO 450 STEP 3
 150  LET t=w*rd: GO SUB 1000
 160  LET x= INT (u+x+h)
 170  LET y=INT (v+y+h)
 180  IF w=90 THEN LET x1=x: LET
y1=y: GO TO 220
 190  LET x2=x: LET y2=y
 200  PLOT x1,y1: DRAW x2-x1,y2-
y1
 210  LET x1=x2: LET y1=y2
 220 NEXT w
 230 IF  INKEY$="" THEN GO TO 23
0
 240 STOP
 250:
1000 LET x=k*SIN (2*t)*(2.5+COS
(3*t))
1010 LET y=k*2*COS (3*t)
1020 RETURN
```



Het tekenen van de x- en y-as is achterwege gelaten om de 'kop' beter te laten uitkomen.

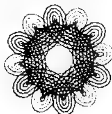
Voor een echte vliegenkop met 'ogen' moet u in de appendix kijken.

Hoe geraffineerder u de functies x = f(t) en y = f(t) kiest, des te ongewoner worden de figuren. Probeer hier hoe creatief u kunt zijn.

De 'huiswiskundigen' bij computerfabrikanten hebben vaak hun eigen lievelingsfuncties. Die zie je dan ook vaak in een demonstratieprogramma optreden.

Programma 20 tekent een stelsel krommen. Hewlett-Packard publiceerde deze tekening enige tijd geleden. Onder het programma staat een tabel met waarden voor a en b, die heel mooie figuren opleveren. Probeer zelf andere combinaties. U zult verrukt zijn over wat u ziet!

a=-6/b=1



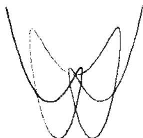
```
 100 REM programma 20 vlinders
 110 CLS
 120 LET u=128: LET v=87
 130 LET h=0.5: LET rd=PI/180
 140 LET kx=8: LET ky=6
 150 INPUT "Toets a en b in  ";a
,b
 160 CLS
 170 FOR n=-3 TO 3
 180  FOR w=0 TO 360 STEP 1
 190   LET t=w*rd
 200   LET x=(a+b)*COS (t)-n*b*C
OS ((a+b)/b*t)
 210   LET y=(a+b)*SIN (t)-n*b*S
IN ((a+b)/b*t)
 220   LET xx=INT (u+kx*x+h)
 230   LET yy=INT (v+ky*y+h)
 240   IF w=0 THEN LET x1=xx: LE
T y1=yy: GO TO 280
 250   LET x2=xx: LET y2=yy
 260   PLOT x1,y1: DRAW x2-x1,y2
-y1
 270   LET x1=x2: LET y1=y2
 280  NEXT w
 290 NEXT n
 300 IF INKEY$="" THEN GO TO 300
 310 STOP
```

Tabel voor mooie figuren

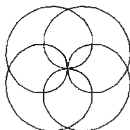
a	−6	−6	4	2	6	4,5	....
b	1	2	1	2	1	1,5	....

Kies in deze programma's de stapgrootte voor hoek w (FOR w=.....) gerust 1; dit duurt wat langer maar geeft mooiere tekeningen.

Het laatste programma uit dit hoofdstuk tekent 'supersymmetrische' figuren. Ook nu geven we een tabel met waarden voor de parameters a, b en c. Het is ongelooflijk hoe de figuur totaal verandert als we andere waarden voor één of meer parameters kiezen. Het idee voor programma 21 is afkomstig uit het Amerikaanse tijdschrift Creative Computing.



a=2/b=7/c=3



a=6/b=6/c=4

```
 100 REM programma 21 symmetrisc
he krommen
 110 CLS
 120 INPUT "Toets a,b,c in  ";a,
b,c
 130 LET u=128: LET v=88
 140 LET h=0.5: LET rd=PI/180: L
ET k=87
 150 CLS
 160 FOR w=0 TO 360
 170  LET t=w*rd: LET r=k*SIN (c
*t)
 180  LET x2=INT (u+r*COS (a*t)+
h)
 190  LET y2=INT (v+r*SIN (b*t)+
h)
 200  IF w=0 THEN LET x1=x2: LET
 y1=y2: GO TO 230
 210  PLOT x1,y1: DRAW x2-x1,y2-
y1
 220  LET x1=x2: LET y1=y2
 230 NEXT w
 240 IF INKEY$="" THEN GO TO 240
 250 STOP
```

Kijk in de appendix hoe u de waarden voor a, b en c samen met de figuur op het scherm kunt afdrukken.

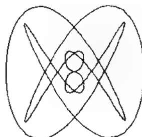
Tabel voor mooie figuren

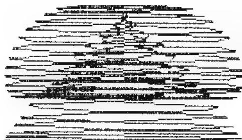
a	2	6	4	1	3	2	....
b	7	6	6	1	3	2	....
c	3	4	1	4	5	9	....

Het zou jammer zijn als u bij het zien van een mooie figuur de waarden voor a, b en c niet hebt genoteerd. Verander het programma zo, dat na afloop van het tekenen de waarden van a, b en c op het beeldscherm of op de printer worden afgedrukt (zie de appendix). Ook heel leuk zijn de combinaties:

a	20	-40	100
b	-1	-40	-0,5
c	3	10	3

a=4/b=6/c=1





a=-1/b=-40/c=-0,5

4 Tekenen van driedimensionale figuren

In dit en het volgende hoofdstuk gaan we ons bezighouden met het tekenen van driedimensionale lichamen (zo heet dat in de wiskunde) zoals kubussen, prisma's, pyramiden, kegels en bollen, en met het tekenen van driedimensionale grafieken van functies. Zo'n driedimensionale grafiek is een vlak in de ruimte met een vergelijking van de vorm $z = f(x,y)$. Zo krijgen we de bekende 'Mexicaanse hoed' (zie p.73) als we de functie ■ $= e^{-(x^2+y^2)}$ tekenen (e is het grondtal van de natuurlijke logaritme; $e = 2,71828...$).

Er bestaan veel programma's waarmee driedimensionale figuren getekend kunnen worden. Waarom we toch ook in dit boek dergelijke programma's laten zien komt, omdat er aan de meeste van deze programma's drie nadelen kleven:

1. De meeste programma's voor driedimensionaal-tekenen zijn voor een bepaald graphics-systeem ontworpen en zijn slechts met veel inspanning geschikt te maken voor andere systemen.

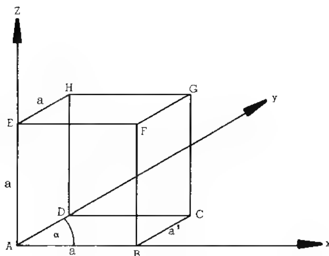
2. De wiskundige theorie die aan het driedimensionaal-tekenen ten grondslag ligt wordt zelden of zeer summier behandeld.

3. Veel programma's zijn uiterst ingenieus ontworpen en draaien op de kleine microcomputers, in BASIC, heel langzaam.

Met de volgende programma's en stukjes theorie hopen we deze drie nadelen uit de weg te ruimen.

Er bestaan verschillende methoden om een driedimensionaal lichaam, bijvoorbeeld een kubus, in een plat vlak te projecteren. Wij hanteren de projectiemethode, die u wellicht op school gebruikt hebt (of nog gebruikt). Stelt u zich een doorzichtige kubus voor met ribben van draadijzer. U staat voor deze kubus een beetje rechts van het

midden en kijkt er zo'n beetje schuin bovenop. Uw gezichtsstralen projecteren elke hoek, met de daaraan verbonden ribben, van de kubus in een, achter de kubus liggend, projectievlak. Zo ontstaat de bekende 'schuine' afbeelding van een kubus.



De verticale en horizontale ribben worden op ware grootte getekend. De ribben die naar achteren lopen worden verkort weergegeven.

a' = k.a k = verkleiningsfactor

De rechte hoek tussen de ribben BA en AD in het grondvlak lijkt eveneens kleiner te zijn geworden (zie α in de bovenstaande figuur). U kunt gemakkelijk nagaan dat door het vastleggen van alpha (bijvoorbeeld 45°) en k (bijvoorbeeld 0,5) de projectierichting eenduidig bepaald is.

Wat we nodig hebben zijn transformatievergelijkingen die voor een bepaalde α en k de coördinaten x,y,z van een punt op de ruimtelijke kubus projecteren op de coördinaten x' en y' van het geprojecteerde punt in het platte (projectie-)vlak. Deze formules vormen de kern van alle volgende programma's.

Afleiding van de transformatievergelijkingen

Het projectievlak is ons beeldscherm. We gebruiken ook nu een schermresolutie van 256 bij 176 punten. De oorsprong van het ruimtelijke coördinatenstelsel (xyz) moet precies in het midden van het beeldscherm geprojecteerd worden. De oorsprong van het beeldschermcoördinatenstelsel ligt in de linkerbenedenhoek. De x'-as wijst naar rechts; de y'-as naar boven.

De afbeelding van het ruimtelijke punt P(x,y,z) op het punt P'(x',y') in het platte vlak komt tot stand met behulp van de volgende transformatievergelijkingen:

$x' = u + x + k.y.\cos\alpha$ èn
$y' = v + (k.y.\sin\alpha + z)$

Hoe we aan deze vergelijkingen komen is uit onderstaande figuur direct (nou ja, direct.....) duidelijk.

175
y'
z
y
P(x,y,z) = p'(x',y')
z
kysinα
ky
α
v=88
y
x
kycosα
0
0
u=128
x' 255

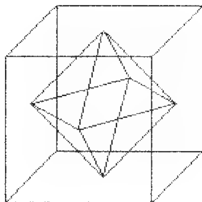
Nemen we c = k.cos(α), s = k.sin(α) en h = 0,5 dan zijn de transformatieformules in BASIC:

```
INT(u+x+c*y+h)     voor de x-coördinaat en
INT(v+s*y+z+h)     voor de y-coördinaat.
```

Meer wiskunde hebben we voor dit hoofdstuk niet nodig.

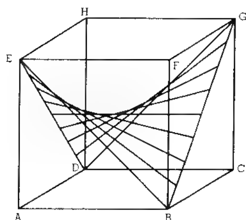
Programma 22 laat zien hoe je van elk, door rechte lijnen begrensd lichaam (polyeder) een projectie kan maken. Hiervoor is het noodzakelijk dat de waarden van de coördinaten x,y,z in alle hoekpunten bekend zijn. Als voorbeeld van het gebruik van dit programma tekenen we een kubus met een ingeschreven achtvlak (oktaëder). De gevolgde programmeertechniek leidt ook bij een viervlak, prisma of pyramide tot het gewenste resultaat.

We nemen aan dat het middelpunt van de kubus samenvalt met de oorsprong van het ruimtelijke coördinatensysteem. Om het programma 'sneller' te maken nemen we de coördinaten van de acht hoekpunten ABCDEFGH van de kubus en de coördinaten van de zes hoekpunten IJKLMN van de achthoek op in DATAregels. Het tekenen van de ribben wordt door de letterstring z$ bestuurd. Zo betekent z$ = "ABBCCDDA" dat de computer van A naar B, van B naar C, van C naar D en van D naar A rechte lijnen moet trekken. Met de CODE-opdracht halen we steeds één letter uit de string z$ en maken er een getal van (A=1, B=2, C=3, ...). Hiermee moet de werking van het programma duidelijk zijn. Kies voor alpha 45° en voor k 0,5; dat geeft de mooiste projectie.

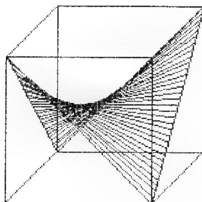


```
 100 REM programma 22
          kubus met 8-vlak
 110 CLS
 120 INPUT "Alpha in graden (45)
   ";a
 130 INPUT "Verkleiningsfactor (
,5)    ";k
 140 LET u=128: LET v=88
 150 LET h=0.5: LET rd=PI/180
 160 LET w=a*rd: LET c=k*COS (w)
: LET s=k*SIN (w)
 170 DIM x(14): DIM y(14): DIM z
$(2,24)
 180 FOR j=1 TO 14
 190   READ x,y,z
 200   LET x(j)=INT (u+x+c*y+h)
 210   LET y(j)=INT (v+s*y+z+h)
 220 NEXT j
 230 CLS
 240 FOR n=1 TO 2
 250   READ z$(n): LET l=LEN (z$(
n))
 260   FOR m=1 TO l-1 STEP 2
 270     LET i=CODE z$(n)(m)-64
 280     LET j=CODE z$(n)(m+1)-64
 290     PLOT x(i),y(i): DRAW x(j)
-x(i),y(j)-y(i)
 300   NEXT m
 310 NEXT n
 320 IF INKEY$="" THEN GO TO 320
 330 STOP
 340:
 350 DATA -60,-60,-60,60,-60,-60
 360 DATA 60,60,-60,-60,60,-60
 370 DATA -60,-60,60,60,-60,60
 380 DATA 60,60,60,-60,60,60
 390 DATA 0,0,-60,0,-60,0,60,0,0
 400 DATA 0,60,0,-60,0,0,0,0,60
 410:
 420 DATA "ABBCCDDAAEBFCGDHEFFGG
HHE"
 430 DATA "IJIKILIMJKKLLMMJJNKNL
NMN"
```

In programma 23 wordt in een kubus een zadelvlak getekend. Het geeft het idee van een gekromd vlak in een kubus.



Bekijk de bovenstaande figuur. De diagonalen BG en ED zijn elk in acht (n) gelijke stukken verdeeld en de zo ontstane punten zijn paarsgewijs door een rechte lijn met elkaar verbonden. Kies in het programma voor n, bijvoorbeeld, de waarde 32. De bovenste kubus hebben wij getekend, de onderste de computer!



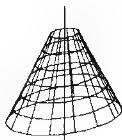
```
 100 REM programma 23
          kubus met zadelvlak
 110 CLS
 120 INPUT "Alpha in graden (45)
   ";a
 130 INPUT "Verkleiningsfactor (
,5)    ";k
 135 INPUT "Hoeveel lijnen (32)
  ";n
 140 LET u=128: LET v=88
 150 LET h=0.5: LET rd=PI/180
 160 LET w=a*rd: LET c=k*COS (w)
: LET s=k*SIN (w)
 170 DIM x(8): DIM y(8): DIM z$(
24)
 180 FOR j=1 TO 8
 190   READ x,y,z
 200   LET x(j)=INT (u+x+c*y+h)
 210   LET y(j)=INT (v+s*y+z+h)
 220 NEXT j
 230 CLS
 250 READ z$: LET l=LEN (z$)
 260 FOR m=1 TO l-1 STEP 2
 270     LET i=CODE z$(m)-64
 280     LET j=CODE z$(m+1)-64
 290     PLOT x(i),y(i): DRAW x(j)
-x(i),y(j)-y(i)
 300 NEXT m
 305 FOR j=0 TO n
 310   LET x1=INT (x(2)+j*(x(7)-x
(2))/n+h)
 315   LET y1=INT (y(2)+j*(y(7)-y
(2))/n+h)
 320   LET x2=INT (x(5)+j*(x(4)-x
(5))/n+h)
 325   LET y2=INT (y(5)+j*(y(4)-y
(5))/n+h)
 330   PLOT x1,y1: DRAW x2-x1,y2-
y1
 335 NEXT j
 340 PLOT x(2),y(2): DRAW x(7)-x
(2),y(7)-y(2)
 345 PLOT x(5),y(5): DRAW x(4)-x
(5),y(4)-y(5)
 347 IF INKEY$="" THEN GO TO 347
 349 STOP
 350 DATA -60,-60,-60,60,-60,-60
 360 DATA 60,60,-60,-60,60,-60
 370 DATA -60,-60,60,60,-60,60
 380 DATA 60,60,60,-60,60,60
 410:
 420 DATA "ABBCCDDAAEBFCGDHEFFGG
HHE"
```

Met programma 24 kunnen we, naar keuze, cilinders, kegels of afgeknotte kegels tekenen. Als we voor r1 en r2 dezelfde waarde invoeren, krijgen we een cilinder. Als r2 kleiner is dan r1 krijgen we een afgeknotte kegel en voor r2 = 0 krijgen we een kegel.

We plaatsen het lichaam zo dat het grondvlak bij z = -60 ligt en het bovenvlak (de top) bij z = +60. Het programma tekent zeven breedtecirkels met een onderlinge afstand van z = 20 en 16 lijnen op de kegel- (of cilinder-) mantel loodrecht op de breedtecirkels. Tot slot wordt verticaal de z-as getekend.

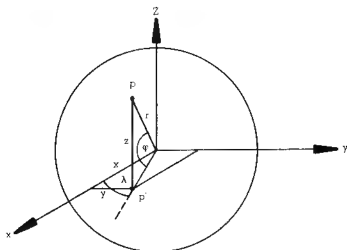
```
 100 REM programma 24
          cylinders,kegels
 110 CLS
 120 INPUT "Alpha in graden (45)
    ";a
 130 INPUT "Verkleiningsfactor (
.5)     ";k
 140 INPUT "Stralen r1 en r2
";r1,r2
 150 LET u=128: LET v=88
 160 LET h=0.5: LET rd=PI/180
 170 LET dr=(r1-r2)/6
 180 LET w=a*rd: LET n=0
 190 LET c=k*COS (w): LET s=k*SI
N (w)
 200 CLS
 210 FOR z=-60 TO 60 STEP 20
 220  LET r=r1-n*dr
 230  FOR w=0 TO 360 STEP 30
 240   LET w1=w*rd
 250   LET x=r*COS (w1): LET y=r
*SIN (w1)
 260   IF w1<>0 THEN GO TO 300
 270   LET x1=INT (u+x+c*y+h)
 280   LET y1=INT (v+s*y+z+h)
 290   GO TO 340
 300   LET x2=INT (u+x+c*y+h)
 310   LET y2=INT (v+s*y+z+h)
 320   PLOT x1,y1: DRAW x2-x1,y2
-y1
 330   LET x1=x2: LET y1=y2
 340  NEXT w
 350  LET n=n+1
 360 NEXT z
 370 FOR w=0 TO 360 STEP 23
 380  LET w1=w*rd
 390  LET x=r1*COS (w1): LET y=r
1*SIN (w1)
 400  LET x1=INT (u+x+c*y+h)
 410  LET y1=INT (v+s*y-60+h)
 420  LET x=r2*COS (w1): LET y=r
2*SIN (w1)
 430  LET x2=INT (u+x+c*y+h)
 440  LET y2=INT (v+s*y+60+h)
 450  PLOT x1,y1: DRAW x2-x1,y2-
y1
 460 NEXT w
 470 PLOT u,0: DRAW 0,175
 480 IF INKEY$="" THEN GO TO 480
 490 STOP
```

Vindt u de breedtecirkels te 'grof' neem dan in regel 230 als stapgrootte van de FOR-lus bijvoorbeeld 3. Het tekenen duurt dan wel een stuk langer!



Erg leuk is het tekenen van een bol met breedtelijnen en meridianen. Zoals bekend is kunnen we elk punt op het boloppervlak eenduidig vastleggen met de straal van de bol (r), de geografische breedte (φ) en de geografische lengte (λ) (zie figuur). Het omzetten van deze bolcoördinaten r,φ,λ in cartesische coördinaten (x,y,z) geschiedt met de volgende drie vergelijkingen:

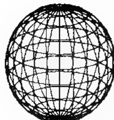
$x = r\cos\varphi\cos\lambda$
$y = r\cos\varphi\sin\lambda$
$z = r\sin\varphi$



De bolvergelijking in cartesische vorm is overigens $x^2+y^2+z^2=r^2$.

In programma 25 wordt de hoek φ door w en de hoek λ door p vertegenwoordigd. Kies voor alpha 90° en voor k 0,5. Andere waarden vervormen de bol.

```
 100 REM programma 25 bol
 110 CLS
 120 INPUT "Alpha in graden (90)
   ";a
 130 INPUT "Verkleiningsfactor (
.5)    ";k
 140 INPUT "Straal, maximaal 78
   ";r
 150 LET u=128: LET v=88
 160 LET h=0.5: LET rd=PI/180
 170 LET w=a*rd
 180 LET c=k*COS (w): LET s=k*SI
N (w)
 190 CLS
 200 FOR w=-90 TO 90 STEP 15
 210  LET w1=w*rd: LET r1=r*COS
(w1)
 220  FOR p=0 TO 360 STEP 3
 230   LET p1=p*rd: LET x=1.15*r
1*COS (p1)
 240   LET y=r1*SIN (p1): LET z=
r*SIN (w1)
 250   IF p=0 THEN LET x1=INT (u
+x+h): LET y1=INT (v+z+h): GO TO
 300
 260   LET x2=INT (u+x+c*y+h)
 270   LET y2=INT (v+s*y+z+h)
 280   PLOT x1,y1: DRAW x2-x1,y2
-y1
 290   LET x1=x2: LET y1=y2
 300  NEXT p
 310 NEXT w
 320 FOR p=0 TO 180 STEP [illegible]
 330  LET p1=p*rd
 340  FOR w=0 TO 360 STEP 3
 350   LET w1=w*rd: LET r1=r*COS
 (w1)
 360   LET x=1.15*r1*COS (p1)
 370   LET y=r1*SIN (p1): LET z=
r*SIN (w1)
 380   IF w=0 THEN LET x1=INT (u
+x+c*y+h): LET y1=INT (v+s*y+z+h
): GO TO 430
 390   LET x2=INT (u+x+c*y+h)
 400   LET y2=INT (v+s*y+z+h)
 410   PLOT x1,y1: DRAW x2-x1,y2
-y1
 420   LET x1=x2: LET y1=y2
 430  NEXT w
 440 NEXT p
 450 IF INKEY$="" THEN GO TO 450
 460 STOP
```



Zie de appendix voor 'mooiere' bollen.

Een geliefd demonstratieprogramma van computerleveranciers is een om zijn as draaiend driedimensionaal lichaam. Meestal wordt als 'draai-as' de z-as gekozen, die dan samenvalt met de lengte-as van het lichaam.

Microcomputers die alleen een BASIC-vertolker bezitten zijn te langzaam om zo'n draaiing real-time uit te voeren. Er zijn namelijk nogal ingewikkelde trigonometrische berekeningen voor nodig, die tamelijk veel tijd in beslag nemen. Daarnaast duurt het tekenen van het lichaam in een bepaalde stand in BASIC zo'n 1 à 2 seconden. Zouden we in een BASIC-programma steeds het lichaam tekenen, uitwissen, draaien, tekenen, uitwissen, draaien, ... enz., dan zien we het draaiende lichaam in plaats van een vloeiende beweging een schokkerige beweging maken. De beste oplossing voor dit probleem is de volgende.

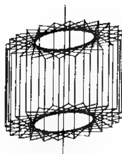
Deel de totale middelpuntshoek van 360° in n even grote hoeken. Voor elk van deze hoeken

$$\omega_1 = \frac{360°}{n},\quad \omega_2 = 2.\omega_1,\ \ldots\ldots,\ \omega_n = n.\omega_1 = 360°$$

berekenen we van te voren de coördinaten van de hoekpunten van het lichaam. Voor elke stand slaan we deze hoekpuntscoördinaten in een array op. Al deze berekeningen voeren we in BASIC uit bij een leeg beeldscherm. Als de berekeningen klaar zijn wordt een machinetaalprogramma uitgevoerd dat de eerste groep coördinaten uit de array haalt, het lichaam tekent, na een bepaalde tijd het beeldscherm wist, de volgende groep coördinaten ophaalt, ... enz. Deze oplossing heeft een redelijk vloeiende draaiing tot gevolg. Dit programma is geschreven op een CBM 3016 die een 6502-microprocessor bezit. Hier doen we echter anders (zie volgende pagina).

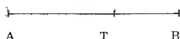
Programma 26 laat een driezijdig prisma om de z-as draaien. De z-as is zowel de lengte-as als de symmetrie-as van het prisma. Het programma tekent een prisma en wacht vervolgens met het draaien en opnieuw tekenen tot we een toets indrukken. Elk volgend prisma wordt over zijn voorgangers heen getekend. We kunnen het draaien dus vertraagd gadeslaan.

```
 100 REM programma 26
          draaiend prisma
 110 CLS
 120 INPUT "Alpha in graden (45)
   ";a
 130 INPUT "Verkleiningsfactor (
0,5)    ";k
 140 INPUT "Draaihoek in graden
(45)    ";om
 150 LET u=128: LET v=88
 160 LET h=0.5: LET rd=PI/180: L
ET r=75
 170 LET w=a*rd: LET c=k*COS (w)
: LET s=k*SIN (w)
 180 CLS
 190 PLOT u,0: DRAW 0,175
 200 DIM x(8): DIM y(8)
 210 FOR w=0 TO 360 STEP om
 220  FOR j=1 TO 4
 230   LET w1=(w+j*120)*rd
 240   LET x=r*COS (w1): LET y=r
*SIN (w1)
 250   LET x(j)=INT (u+x+c*y+h)
 260   LET y(j)=INT (v+s*y-60+h)
 270   LET x(j+4)=x(j)
 280   LET y(j+4)=INT (v+s*y+60+
h)
 290  NEXT j
 300 :
 310  FOR j=1 TO 3
 320   PLOT x(j),y(j): DRAW x(j+
1)-x(j),y(j+1)-y(j)
 330   PLOT x(j),y(j): DRAW x(j+
4)-x(j),y(j+4)-y(j)
 340   PLOT x(j+4),y(j+4): DRAW
x(j+5)-x(j+4),y(j+5)-y(j+4)
 350  NEXT j
 360 IF INKEY$="" THEN GO TO 360
 370 NEXT w
 380 STOP
```



Programma 27 tekent de projectie van een regelmatig twintigvlak (ikosaëder). Het is bekend dat er slechts vijf 'echt-regelmatige' veelvlakken (polyeders) zijn, namelijk een tetraëder (regelmatig drievlak), een kubus (regelmatig zesvlak), een octaëder (regelmatig achtvlak), een dodekaëder (regelmatig twaalfvlak) en een ikosaëder (regelmatig twintigvlak). Een ikosaëder heeft 12 hoeken, 30 ribben en 20 vlakken. Een vlak is een gelijkzijdige driehoek. Als we programma 22 willen gebruiken om zo'n ikosaëder te tekenen dan hebben we de coördinaten van alle 12 hoekpunten nodig. Als we het ikosaëder in een speciale stand zetten kunnen we de hoekpuntcoördinaten met behulp van de techniek van de 'gulden snede' relatief eenvoudig berekenen. Wie zich voor de hierachterliggende wiskundige theorie interesseert moet er maar eens een meetkundeboek op naslaan. De 'gulden-snede'-techniek deelt een lijnstuk AB in twee stukken AT en TB op zo'n manier dat de volgende evenredigheid geldt:

$$AB : AT = AT : TB$$



Kiezen we voor AB de lengte 1, dan kunnen we de lengte van AT berekenen; immers

$$\frac{AB}{AT} = \frac{AT}{TB}, \quad \text{dus} \quad \frac{1}{t} = \frac{t}{1-t}, \quad \text{als } t = AT$$

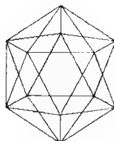
dus dan moet gelden $t^2 = 1-t$ ofwel $t^2+t-1=0$.
Met de alombekende abc-formule berekenen we nu

$$t_{1,2} = \frac{-1 \pm \sqrt{1+4}}{2}$$

De positieve wortel geeft: $t = \dfrac{-1+\sqrt{5}}{2}$

Dit is de lengte van het grootste van de twee stukken die we krijgen als we een lijnstuk, ter lengte 1, volgens de 'gulden snede' in twee stukken verdelen ($t = 0{,}618$; $1-t = 0{,}382$).
Deze waarde wordt in regel 180 berekend en in de regels 220-330 gebruikt om de hoekpuntcoördinaten van het ikosaëder te berekenen. Om te zorgen dat het twintigvlak voldoende groot getekend wordt zijn alle coördinaten met 70 (f) vermenigvuldigd. Kies voor alpha (a) 90° en voor k de waarde 0,4. Andere waarden vertekenen het beeld. Alpha = 180° en k=0,6 geeft trouwens iets onverwachts!

```
 100 REM programma 27 ikosaeder
 110 CLS
 120 INPUT "Alpha in graden (90)
   ";a
 130 INPUT "Verkleiningsfactor (
0.4)     ";k
 140 LET u=128: LET v=88
 150 LET h=0.5: LET rd=PI/180
 160 LET w=a*rd
 170 LET c=k*COS (w): LET s=k*SI
N (w)
 180 LET t=(SQR (5)-1)/2: LET f=
70
 190 CLS
 200 DIM x(12): DIM y(12)
 210 DIM z(12): DIM z$(3,20)
 220 LET x(1)=0: LET y(1)=f*t: L
ET z(1)=-f
 230 LET x(2)=0: LET y(2)=-f*t:
LET z(2)=-f
 240 LET x(3)=f: LET y(3)=0: LET
 z(3)=-f*t
 250 LET x(4)=-f: LET y(4)=0: LE
T z(4)=-f*t
 260 LET x(5)=f*t: LET y(5)=f: L
ET z(5)=0
 270 LET x(6)=-f*t: LET y(6)=f:
LET z(6)=0
 280 LET x(7)=f*t: LET y(7)=-f:
LET z(7)=0
 290 LET x(8)=-f*t: LET y(8)=-f:
 LET z(8)=0
 300 LET x(9)=f: LET y(9)=0: LET
 z(9)=f*t
 310 LET x(10)=-f: LET y(10)=0:
LET z(10)=f*t
 320 LET x(11)=0: LET y(11)=f*t:
 LET z(11)=f
 330 LET x(12)=0: LET y(12)=-f*t
: LET z(12)=f
 340 FOR n=1 TO 3
 350  READ z$(n): LET l=LEN (z$(
n))
 360  FOR m=1 TO l-1 STEP 2
 370   LET i=CODE z$(n)(m)-64
 380   LET j=CODE z$(n)(m+1)-64
 390   LET x1=INT (u+x(i)+c*y(i)
+h)
 400   LET y1=INT (v+s*y(i)+z(i)
+h)
 410   LET x2=INT (u+x(j)+c*y(j)
+h)
 420   LET y2=INT (v+s*y(j)+z(j)
+h)
 430   PLOT x1,y1: DRAW x2-x1,y2
-y1
 440  NEXT m
 450 NEXT n
 460 IF INKEY$="" THEN GO TO 460
 470 STOP
 480:
 490 DATA "BCCEEFFDDBABACAEAFAD"
 500 DATA "BHHDDJJFFKKEEIICCGGB"
 510 DATA "GHHJJKKIIGLGLHLJLKLI"
```



5 Het tekenen van vlakken in de ruimte

In het vorige hoofdstuk hebben we ons uitvoerig beziggehouden met het tekenen van driedimensionale lichamen. We herhalen hiervan nog eens de belangrijkste punten:

1. We gebruiken de parallelprojectie om een driedimensionaal lichaam met n hoekpunten P(x,y,z) in een plat vlak te tekenen. Voor de projectiehoek alpha (α) gebruiken we bij voorkeur de waarde 45° of 60°. De schuin-naar-achteren lopende ribben worden korter getekend dan ze in werkelijkheid zijn. Als verkleiningsfactor kiezen we vaak 1/2 of 1/3.

2. Elk punt P(x,y,z) van het ruimtelijke lichaam wordt op een punt P'(x',y') in het projectievlak geprojecteerd. De hierbij gebruikte projectieformules zijn:

   $x' = u + x + k.y.\cos\alpha$
   $y' = v + k.y.\sin\alpha + z$

   Zoals gebruikelijk zijn u en v de coördinaten van het midden van het beeldscherm. Tot nu toe hebben we u = 128 en v = 88 verondersteld.

In de volgende BASIC-programma's zien we de bovenstaande transformatievergelijkingen in de vorm:

```
xg = INT(u+xx+c*yy+h)
yg = INT(v+s*yy+z+h)
```

De betekenis van de gebruikte variabelen is:

xg,yg	: beeldschermcoördinaten x',y'
xx,yy	: de lopende coördinaten x,y
c	: k*COS(w*rd)
s	: k*SIN(w*rd)
w	: projectiehoek in graden

rd : factor π/180 voor omrekenen van graden in radialen
k : verkleiningsfactor

Tekenen van driedimensionale functies

Als paradepaardje van menige demonstratie van Hoge-Resolutie-Graphics wordt vaak een programma gebruikt dat een of andere fraaie grafiek van een driedimensionale functie tekent. Als leek vraag je je af hoe ze weten welke functies ze moeten nemen, hoe het tekenen van de grafiek van zo'n functie geprogrammeerd wordt, en hoe ze het programmatechnisch voor elkaar krijgen dat de 'onzichtbare lijnen' ook inderdaad niet getekend worden. Bij dergelijke demonstraties wordt doorgaans geen programmalisting getoond. Mocht dit wel het geval zijn dan is het programma vaak zo machine-afhankelijk dat het omzetten van het programma naar een andere machine lastig, zo niet ondoenlijk is.

In dit hoofdstuk hopen we u antwoord te kunnen geven op de volgende vragen:

1. Hoe vind ik 'mooie' driedimensionale functies?

2. Hoe ziet een algemeen programma voor het tekenen van een dergelijke functie eruit?

3. Hoe onderdruk je het tekenen van de onzichtbare lijnen (hidden lines)?

We beginnen met de definitie van een driedimensionale functie.

Elke functie van de vorm ■ = f(x,y) heet een driedimensionale functie en vormt een, meestal, gekromd vlak in een driedimensionaal (ruimtelijk) coördinatenstelsel. Speciaal voor niet-wiskundigen volgt nu een voorbeeld van een functie z = f(x,y) en het gekromde vlak dat als 'grafiek' bij de functie hoort.

Stel dat we voor ■ = f(x,y) de volgende vergelijking nemen:

$$z = e^{-(x^2+y^2)}$$ (e is het grondtal van de natuurlijke logaritme; e = 2,718284183...)

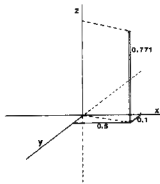
Voor elk punt (x,y) in het (platte) X-Y-vlak kunnen we nu de daarbij horende waarde van z uitrekenen. Als x = 0,5 en y = 0,1 dan is

$z = e^{-(0,25+0,01)} \Rightarrow$

$z = e^{-0,26} \Rightarrow$

z = 0,771, want 2,71828... tot de macht -0,26 is 0,771

Zet nu (in gedachten) in het voetpunt P(0,5;0,1) in het X-Y-vlak een staafje met een lengte van 0,771 rechtop. Doe ditzelfde voor nog een groot aantal punten (x,y).



Leg nu over al deze staafjes een elastische folie. Dit (golvende) stuk folie vormt een goed model van het vlak met vergelijking $z = e^{-(x^2+y^2)}$.
Hoe dit eruit zou zien kunt u zien op p.73.

Antwoord op de eerste vraag

Zoek met behulp van programma 7 uit hoofdstuk 2 een willekeurige contine functie die symmetrisch ten opzichte van de y-as is. De grafiek van de functie moet 'bergen en dalen' vertonen (de functie moet maxima en minima hebben). Erg geschikt zijn trigonometrische functies (sin, cos) en combinaties van deze functies. Ook geschikt zijn functies waarin alleen termen voorkomen met 'even-machten' van x en exponentiële functies.

Enkele voorbeelden:

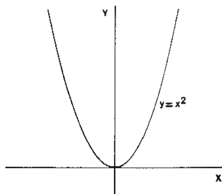
$y = e^{-x^2}$; $y = \sin(x)$

$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}$$

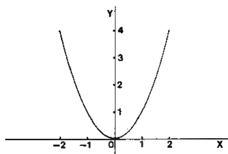
Bekijk nu de grafiek van zo'n functie in het interval $-a \leq x \leq a$. Dit 'stuk grafiek gaan we draaien rond de y-as. Zo ontstaat een, ten opzichte van de y-as, draaisymmetrisch driedimensionaal vlak. Als we nu de y-as als z-as kiezen, dan wordt de vergelijking van een dergelijk draaisymmetrisch ruimtelijk vlak (rond de z-as) van de vorm

$$z = f(r) \qquad \text{met} \quad r = \sqrt{x^2+y^2}$$

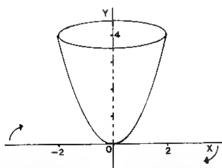
We zullen dit aan een eenvoudig voorbeeld proberen te verklaren. We kiezen als voorbeeld de eenvoudige paraboolvergelijking $y=x^2$.



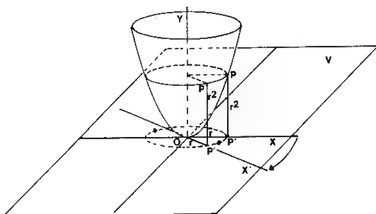
Stel we bekijken de grafiek voor $-2 \leq x \leq 2$:



Als we nu de x-as rond de y-as laten draaien (de x-as komt als het ware loodrecht het papier uit), dan ontstaat een draaisymmetrisch vlak rond de y-as. Dit is een kegel:



Elk punt P op de kegelmantel heeft de eigenschap $y=r^2$, waarbij r de afstand is van het geprojecteerde punt P' tot het punt O. We kunnen het vlak V dat door de ronddraaiende x-as wordt gevormd als volgt tekenen:

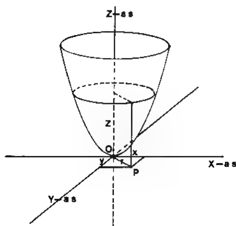


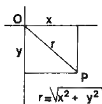
Welnu, als we nu de z-as als y-as nemen en we nemen de y-as in vlak V loodrecht op de x-as, dan zien we dat de afstand r geschreven kan worden als

$r = \sqrt{x^2+y^2}$ (stelling van Pythagoras, zie rechts in de volgende tekening)

en nu wordt $y=r^2$ geschreven als $z=r^2$ met $r^2 = x^2+y^2$, dus de vergelijking van de paraboloïde (kegel) wordt dan

$$z = x^2+y^2$$





Dus $y=x^2$ wordt bij draaiing om de y-as en bij een z-as die de plaats van de y-as inneemt ■ $= x^2+y^2$.
Voor de bovengenoemde functies geldt het volgende:

$y = e^{-x^2}$ wordt $z = e^{-(x^2+y^2)}$

$y = \dfrac{\sin(x)}{x}$ wordt $z = \dfrac{\sin(r)}{r}$

$y = \cos(x) - \dfrac{\cos(3x)}{3} + \dfrac{\cos(5x)}{5} - \dfrac{\cos(7x)}{7}$ wordt

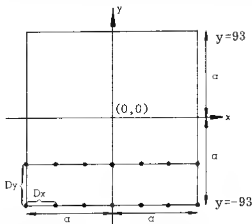
$y = \cos(r) - \dfrac{\cos(3r)}{3} + \dfrac{\cos(5r)}{5} - \dfrac{\cos(7r)}{7}$

met $r = \sqrt{x^2+y^2}$

Voor deze klasse van draaisymmetrische figuren ontwikkelen we een algemeen tekenprogramma.

Antwoord op de tweede vraag

Bekijk de onderstaande tekening. U kijkt recht bovenop het ruimtelijke vlak van een driedimensionale functie. We willen dat het op het grondvlak geprojecteerde vlak een vierkant is met zijden van 2a. Straks zullen we zien dat, als we een mooie tekening van zo'n ruimtelijk vlak op het beeldscherm willen maken, a=93 een heel goede keus is.



Als 'doorgewinterde' programmeur ziet u natuurlijk direct dat het programma voor het tekenen van het vlak een geneste FOR-NEXT (dx,dy) zal bevatten! Als we een vlak als grafiek van z=f(x,y) willen tekenen, moeten we namelijk voor een aantal combinaties van x en y de functiewaarde z = f(x,y) berekenen. We beginnen met y = -93 en laten x met stapjes dx (bijvoorbeeld dx=3) van -93 maar +93 lopen. Voor elke combinatie x,y met y = -93 berekenen we de functiewaarde z = f(x,y). Zo ontstaan de punten (x,y,z) van het ruimtelijke vlak waarvan de geprojecteerde punten (x,y) in de bovenstaande tekening op de onderste zijde van het vierkant als bolletjes getekend zijn. Nu verhogen we y met dy en laten x weer van -93 tot +93 lopen. Dit geeft de tweede reep van het vlak. Op deze wijze ontstaat het hele vlak, waarvan de punten (x,y,z) natuurlijk nog getransformeerd moeten worden naar beeldschermcoördinaten (x',y').

Een eerste globale opzet voor het tekenen van een 'vierkant'-stuk vlak is:

```
FOR y=-93 TO 93 STEP dy
  FOR x=-93 TO 93 STEP dx
    GO SUB 1000
    :
  NEXT x
NEXT y
```

In de subroutine 1000 wordt voor een punt (x,y) de waarde van ■ = f(x,y) berekend. Met de bekende transformaties wordt vervolgens het punt P(x,y,z) overgebracht naar een punt P'(x',y') op het beeldscherm. Nu kan het berekende punt P(x,y,z), dat in het vlak ligt, als het punt P'(x',y') op het scherm worden getekend.

Jammer genoeg lenen niet alle driedimensionale functies zich voor de intervallen $-93 \leq x \leq 93$ en $-93 \leq y \leq 93$; vandaar dat de waarde voor a met een INPUT-opdracht ingelezen wordt. Als we in het programma toch $-93 \leq xx \leq 93$ en $-93 \leq yy \leq 93$ kiezen, moet de ingetoetste waarde a als volgt gebruikt worden:

$$x = xx \cdot \frac{a}{93} \qquad \text{en} \qquad y = yy \cdot \frac{a}{93}$$

dat wil zeggen er geldt: $-a \leq x \leq a$ en $-a \leq y \leq a$.

De z-waarden van met name de trigonometrische driedimensionale functies zal tussen -1 en 1 liggen. Om deze waarden wat 'op te blazen' kan een vermenigvuldigingsfactor (k1) worden ingetoetst. Goede waarden voor k1 liggen tussen 30 en 80.

We kunnen ons programma nu als volgt opschrijven:

begin tekenprogramma voor de functie ■ = f(x,y)		
maak beeldscherm schoon		
lees w, k, a en k1 in		
geef u, v, h, rd, c, enz. een waarde		
maak beeldscherm schoon		
voor y is -93 tot +93 in stapjes dy doe		
	voor x = -93 tot +93 in stapjes dx doe	
		bereken de functiewaarde z = f(x,y)
		bereken de beeldschermcoördinaten xg en yg
		teken het punt (xg,yg) op het beeldscherm
wacht tot een toets wordt ingedrukt		
einde tekenprogramma voor de functie ■ = f(x,y)		

Zoals beloofd laten we nu zien waarom de intervallen $-93 \leq xx \leq 93$ en $-93 \leq yy \leq 93$ zo geschikt zijn om driedimensionale functies te tekenen met een hoog oplossend vermogen. We willen graag de fraaie projectie met $\alpha=45°$ en k=0,5 gebruiken. Wat worden dan de beeldschermcoördinaten xg en yg voor de hoekpunten 'linksonder' en 'rechtsboven' van het vierkant op p.69? Deze punten bepalen immers of de hele grafiek op ons beeldscherm van 256 bij 176 puntjes getekend kan worden. Voor k1 nemen we 50, dat ligt zo'n beetje tussen 30 en 80! Nemen we voor de z-waarde ook 50, dan gaat het punt (x,y,z) met coördinaten (-93,-93,50) over in het beeldschermpunt (xg,yg) met

xg = INT(128-93-0,5.93.cos(45)+0,5) ⇒ 2
yg = INT(88-0,5.93.sin(45)+50+0,5) ⇒ 105

Deze waarden liggen inderdaad binnen ons 'graphic-scherm' ($0 \leq xg \leq 255$ en $0 \leq yg \leq 176$) en de 'breedte' (256) van het scherm wordt heel goed benut, kijk maar naar de coördinaten (xg,yg) voor het punt (93,93,50) rechtsboven:

xg = INT(128+93+0,5.93.cos(45)+0,5) ⇒ 254
yg = INT(88+0,5.93.sin(45)+50+0,5) ⇒ 171

We benutten dus haast de hele schermbreedte ($2 \leq xg \leq 254$) voor het tekenen van de driedimensionale figuur. In programma 28 (regel 270) en in programma 29 (regel 310) is rekening gehouden met eventuele negatieve xg-waarden en xg-waarden groter dan 255. Als dit voorkomt, worden deze waarden op respectievelijk 0 en 255 gezet, zodat het programma niet door een foutmelding afbreekt.

Op de onder- en bovenrand (yy=-93 en yy=+93) gelden vaak heel kleine z-waarden, zodat bij $\alpha=45°$, k=0,5 en z=0 de grafiek op het scherm zal liggen tussen

↓ z-waarde

yg = INT(88+0,5.93.sin(45)+0+0,5) ⇒ 121
en yg = INT(88-0,5.93.sin(45)+0+0,5) ⇒ 55

Ons tekenvlak is dus ongeveer $2 \leq xg \leq 254$ èn $55 \leq yg \leq 121$

In het bovenstaande programmavoorstel wordt gebruik gemaakt van 'puntgraphics'. Hierbij worden de punten puntje voor puntje getekend. Om een enigszins acceptabele tekening te krijgen moeten erg veel 'puntjes' berekend worden (dx=1,dy=1). Dit duurt in BASIC (geneste lussen en vele trigonometrische berekeningen) erg lang. Voor de onderstaande tekeningen betekent dit al snel een tekentijd van meer dan 30 minuten, soms wel een uur. Dit is weinig bevredi-

gend! Veel sneller gaat het als we de 'vectorgraphics'-methode gebruiken. Deze techniek hebben we in alle voorgaande programma's gebruikt; we verbinden steeds twee naast elkaar liggende punten door een recht lijntje. Hierbij hoeven we lang niet zoveel punten te berekenen als bij de 'puntgraphics'-techniek. Bovendien ziet de tekening er beter uit.

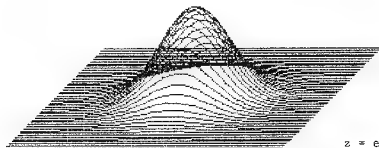
Als antwoord op de tweede vraag komen we dan met het volgende algemene programma voor het tekenen van draaisymmetrische ruimtelijke vlakken. In programma 28, en ook in programma 29, zien we voor xx de variabele q en voor yy de variabele p*.

```
 100 REM programma 28 grafiek va
n z=f(x,y)
 110 CLS
 120 INPUT "Alpha in graden (45
- 135)    ";w
 130 INPUT "Verkleiningsfactor (
.5 - .75)    ";k
 140 INPUT "Rechtergrens voor x
(>0)    ";a
 150 INPUT "Vergrotingsfactor (3
0 - ■0)    ";k1
 160 LET u=128: LET v=88
 170 LET h=0.5: LET rd=PI/180
 180 LET c=k*COS (w*rd): LET s=k
*SIN (w*rd)
 190 LET dx=3: LET dy=5: LET af=
a/93
 200 CLS
 210 FOR p=-93 TO 93 STEP dy
 220  LET y=p*af
 230  FOR q=-93 TO 93 STEP dx
 240   LET x=q*af: GO SUB 1000
 250   LET xg=INT (u+q+c*p+h)
 260   LET yg=INT (v+s*p+z+h)
 270   IF xg<0 THEN LET xg=0
 275   IF xg>255 THEN LET xg=255
 280   IF yg<0 THEN LET yg=0
 285   IF yg>175 THEN LET yg=175
 290   IF q=-9■ THEN LET x1=xg:
LET y1=yg: GO TO 330
 300   LET x2=xg: LET y2=yg
 310   PLOT x1,y1: DRAW x2-x1,y2
-y1
 320   LET x1=x2: LET y1=y2
 330  NEXT q
 340 NEXT p
 350 IF INKEY$="" THEN GO TO 350
 360 STOP
1000 LET z=k1*EXP (-x*x-y*y)
1100 RETURN
```

Met dit programma maakt u de tekening op p.73. Kies op uw ZX Spectrum de volgende waarden: w=45°, k=0,5, a=3 en k1=70. U krijgt dan het mooie 45°-perspectief.

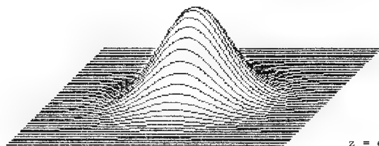
---

*In een FOR-NEXT-lus mogen alleen éénlettergrepige lusvariabelen voorkomen, vandaar q en p in plaats van respectievelijk xx en yy.



$z = e^{-(x^2+y^2)}$

U zult over de bovenstaande tekening niet tevreden zijn. De voorgrond is goed, maar in het achterste stuk zijn ook alle lijnen die voor ons onzichtbaar zijn getrokken. In onderstaande tekening zien we het resultaat met dezelfde waarden voor w, k, a en k1, maar waarin de niet-zichtbare lijnen ook niet getekend zijn. Hoe krijgen we dit voor elkaar?



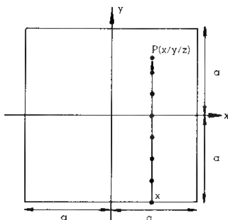
$z = e^{-(x^2+y^2)}$

Antwoord op de derde vraag

Bekijk de onderstaande tekening. Een punt P(x,y,z) van het te tekenen vlak is alleen dan zichtbaar wanneer:

a. het punt 'hoger' ligt dan alle voorgaande punten met dezelfde x-coördinaat of
b. het punt 'lager' ligt dan alle voorgaande punten met dezelfde x-coördinaat.

Vanzelfsprekend geldt dit ook voor de op het beeldscherm geprojecteerde punten P(xg,yg).



We gaan als volgt te werk. We gebruiken twee arrays (h1(xg) en h2(xg)) waarin we voor alle xg-coördinaten in ons vlak de respectievelijk kleinste en grootste yg-waarde bewaren. Vinden we voor een punt met een bepaalde xg-waarde een bijbehorende yg-waarde die óf kleiner is dan h1(xg) óf groter is dan h2(xg), dan is het punt zichtbaar en wordt h1(xg) of h2(xg) gelijk gemaakt aan deze nieuwe kleinste of grootste yg-waarde. Geldt voor een bepaalde xg dat h1(xg)<yg<h2(xg), dan is het punt (xg,yg) niet zichtbaar. h1(xg) en h2(xg) zijn te zien als twee horizonnen.

Als voor een bepaald punt $P_1(x,y,z)$ uit het vlak het beeldpunt P1(xg,yg) berekend is, gaan we kijken of yg kleiner dan of gelijk aan h1(xg) is (yg<=h1(xg)). Is dit zo dan is het punt P1 op het beeldscherm zichtbaar. Nu zetten we de vlag f1 op 1 en we maken h1(xg) gelijk aan yg (h1(xg)=yg). Is yg groter dan h1(xg) (yg>h1(xg)) dan is P1 onzichtbaar en blijft de vlag f1 op 0 staan.

Nu gaan we naar het volgende punt $P_2(x,y,z)$ uit het vlak (we verhogen x met dx). Opnieuw berekenen we het beeldpunt P2(xg,yg). Weer wordt gekeken of yg<=h1(xg) en zonodig wordt de vlag f2 op 1 gezet. De punten P1 en P2 worden alleen dan door een rechte lijn (lijntje) verbonden als beide vlaggen f1 en f2 de waarde 1 hebben, dus als f1xf2=1.

Ditzelfde doen we voor de tweede 'horizon' h2(xg). We kijken of yg>=h2(xg), ..., enzovoorts.

Omdat we bij een vaste y-waarde de x-waarde steeds met dx ophogen (bijvoorbeeld dx=3) en hiermee steeds twee buurpunten P1(xg,yg) en P2(xg,yg) berekenen slaan we als het ware een aantal punten over waarvoor geen h1(xg)- en h2(xg)-waarden berekend

worden. Door lineaire interpolatie tussen de horizonwaarden (h1(xg) en h2(xg)) van twee buurpunten zouden we de horizonwaarden voor de hiertussenliggende punten kunnen berekenen. We hebben deze waarden wel nodig, omdat het kan voorkomen dat, als we y met dy verhogen, we xg-waarden vinden waarvoor nog geen h1(xg)- en h2(xg)-waarden berekend zijn.

Een programma dat de hierboven geschetste werkwijze zou volgen zou, in BASIC geschreven, veel te langzaam zijn. Om de tekensnelheid acceptabel te houden, dat wil zeggen niet langer dan 10 minuten tekenen voor één figuur, passen we de volgende vereenvoudigingen toe:

1. We berekenen alleen de bovenste horizon, die we h(xg) noemen.

2. We passen geen lineaire interpolatie toe bij het berekenen van de array-waarden in h(xg). Alle beeldpunten in een interval ter lengte dx krijgen dezelfde h(xg)-waarde.

Als we de stapgrootte dx maar klein genoeg kiezen (een goede waarde is dx=3), krijgen we ondanks deze twee simplificaties toch acceptabele tekeningen. Hieronder volgt een korte uitleg van het programma.

De regels

```
200 DIM h(256)
210 FOR l=1 TO 256
220 LET h(l)=-1000
230 NEXT l
```

zorgen ervoor dat, voor het tekenen begint, het hele scherm zichtbaar is. Dit is zo als de onderrand van het scherm de horizon is. De waarde -1000 geeft in feite een horizon die ver onder het beeldscherm ligt. De array h bevat in principe voor elke pixel op de horizontale beeldschermlijnen (x-waarden) een horizonwaarde. Als we dx=1 nemen, hebben we inderdaad 256 van deze horizonwaarden nodig. In programma 29 gebruiken we dx=3, hetgeen betekent dat we op de horizontale beeldlijnen steeds 2 pixels overslaan, waarvoor dus ook eigenlijk geen horizonwaarden bekend hoeven te zijn. In feite hebben steeds drie naast elkaar liggende x-pixels dezelfde horizonwaarde. We laten dit aan de hand van een voorbeeld zien.

Stel dat in programma 29 voor een bepaald punt P(x,y,z) berekend is dat dit punt geprojecteerd wordt op het punt met beeldschermcoördinaten xg=40 en yg=126 en laten we aannemen dat dit berekend is voor een waarde xx>-93 (in het programma voor q>-93). Na het berekenen van xg en yg (regels 290 en 300) komt het programma bij regel 330. Omdat we even aannemen dat xx(q)>-93 gaat het pro-

gramma verder met regel 370 (door GO TO 370 in regel 330). De regels 370 en 380 luiden:

```
370 LET f2=0: LET l=INT(xg/dx)+1
380 IF yg>h(l) THEN LET f2=1: LET h(l)=yg
```

De vlag f2 wordt eerst op nul gezet. Vervolgens wordt de index l voor de horizonarray bepaald. Voor xg=40 en dx=3 vinden we l = INT(40/3)+1). Dit geeft l=14. In feite zouden xg=39 en xg=41 ook l=14 opleveren. Steeds leveren drie punten op de x-as dus dezelfde horizonwaarde op. Dit komt omdat dx=3 gekozen is. Nemen we dx=1 dan levert elke xg-waarde tussen 0 en 255 een andere l-waarde op. Voor xg=0 en dx=1 krijgen we l=INT(0/1)+1). Dit geeft l=1; het eerste horizonarray-element. Voor xg=255 en dx=1 vinden we l= INT(255/1)+1, dus l=256. Dit is het grootste horizonarray-element. Voor dx=1 gebruiken we dus alle horizonarray-elementen. Terug naar ons voorbeeld. De waarde voor l is dus 14 als xg=40, yg=126 en dx=3. In regel 380 wordt vervolgens bekenen of de berekende yg-waarde groter is dan de op dat moment geldende horizonwaarde bij xg=40. Is yg inderdaad groter dan die horizonwaarde (yg>h(l) is waar), dan is het punt (xg,yg) zichtbaar en wordt de nieuwe horizonwaarde bij xg=40 gelijk aan de waarde yg. Stel dat de horizonwaarde h(14) gelijk was aan 130. Regel 380 test dan:

```
IF 126>130 THEN LET f2=1: h(14)=126
```

maar 126 is kleiner dan 130, dus blijft 130 de horizonwaarde bij xg=40 en de vlag f2 wordt niet op 1 gezet maar blijft 0. Dit betekent dat het punt (xg=40, yg=126) een onzichtbaar punt is en dus wordt er geen lijn getrokken uit het laatst getekende punt naar dit zojuist berekende punt. In regel 410 wordt hierdoor f1 ook weer nul (f1= f2), zodat we eerst twee keer een f2-waarde van 1 moeten hebben om weer een zichtbaar lijntje te kunnen trekken. Op deze manier ontstaat een tekening met verborgen lijnen (hidden lines), waarvan een aantal na het programma is afgedrukt. In programma 29 zien we de variabelen q en p de rol van xx en yy vervullen.

Met w=45°, k=0,5, a=3 en k1=70 krijgt u de grafiek van $z=e^{-(x^2+y^2)}$ zoals die een paar pagina's eerder is getekend; de niet-zichtbare punten die achter de hoed liggen zijn inderdaad niet te zien!

Hoe kleiner we dx en dy maken, des te gedetailleerder zal de figuur worden. Maar ook 'des te langer zal het tekenen duren'. De combinatie dx=3 en dy=5 is een goed compromis tussen de tekensnelheid en de mate van gedetailleerdheid.

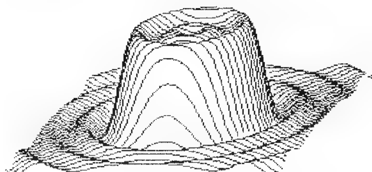
```
 100 REM programma 29
         grafiek van z=f(x,y)
         hidden lines
 110 CLS
 120 INPUT "Alpha in graden (45
- 135)     ";w
 130 INPUT "Verkleiningsfactor (
.5 - .75)    ";k
 140 INPUT "Rechtergrens voor x
(>0)     ";a
 150 INPUT "Vergrotingsfactor (3
0 - 80)     ";k1
 160 LET u=128: LET v=88
 170 LET h=0.5: LET rd=PI/180
 180 LET c=k*COS (w*rd): LET s=k
*SIN (w*rd)
 190 LET dx=3: LET dy=5: LET af=
a/93
 200 DIM h(256)
 210 FOR l=1 TO 256
 220  LET h(l)=-1000
 230 NEXT l
 240 CLS
 250 FOR p=-93 TO 93 STEP dy
 260  LET y=p*af
 270  FOR q=-93 TO 93 STEP dx
 280   LET x=q*af: GO SUB 1000
 290   LET xg=INT (u+q+c*p+h)
 300   LET yg=INT (v+s*p+z+h)
 310   IF xg<0 THEN LET xg=0
 315   IF xg>255 THEN LET xg=255
 320   IF yg<0 OR yg>175 THEN PR
INT "Foute k1 ": STOP
 330   IF q>-93 THEN GO TO 370
 340   LET f1=0: LET l=INT (xg/d
x)+1
 350   IF yg>h(l) THEN LET f1=1:
 LET h(l)=yg
 360   LET x1=xg: LET y1=yg: GO
TO 420
 370   LET f2=0: LET l=INT (xg/d
x)+1
 380   IF yg>h(l) THEN LET f2=1:
 LET h(l)=yg
 390   LET x2=xg: LET y2=yg
 400   IF f1*f2=1 THEN PLOT x1,y
1: DRAW x2-x1,y2-y1
 410   LET x1=x2: LET y1=y2: LET
 f1=f2
 420  NEXT q
 430 NEXT p
 440 IF INKEY$="" THEN GO TO 440
 450 STOP
1000 LET z=k1*EXP (-x*x-y*y)
1010 RETURN
```

Dit programma is een algemeen programma voor het tekenen van draaisymmetrische ruimtelijke vlakken. Als u andere figuren wilt maken, verander dan alleen de functie in de subroutine 1000 en kies mogelijk andere waarden voor de variabelen.

Nu volgt een aantal voorbeelden.

```
1000 LET r=SQR(x*x+y*y)*rd
1010 LET z=k1*(COS(r)-COS(3*r)/3+COS(5*r)/5-COS(7*r)/7
1100 RETURN
```

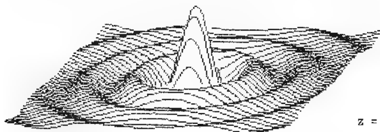
U kiest: w=45, k=0,5, a=180°, k1=35.



$$z = \cos(r) - \frac{\cos(3r)}{3} + \frac{\cos(5r)}{5} - \frac{\cos(7r)}{7}$$

```
1000 LET r=SQR(x*x+y*y)*rd
1010 IF r=0 THEN LET z=k1: RETURN
1020 LET z=k1*SIN(r)/r
1100 RETURN
```

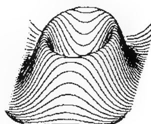
U kiest: w=45, k=0,5, a=1080, k1=50.



$$z = \frac{\sin r}{r}$$

```
1000 LET r=SQR(x*x+y*y)
1010 LET z=k1*EXP(-COS(r/16))
1100 RETURN
```

U kiest: w=45, k=0,5, a=90, k1=30.

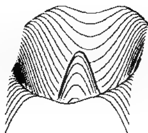


$z = \exp(-\cos(\frac{r}{16}))$

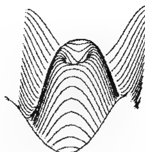
Voor de volgende twee tekeningen geldt:

```
1000 LET r=SQR(x*x+y*y)
1010 LET z=k1*COS(r/16)  respect.  LET z=k1*SIN(r/16)
1100 RETURN
```

U kiest: w=45, k=0,5, a=90, k1=50.



$z = \cos(\frac{r}{16})$



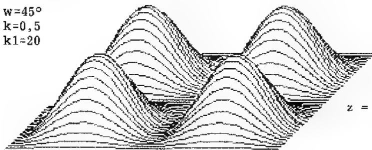
$z = \sin(\frac{r}{16})$

Met dit programma hebben we talrijke andere grafieken gemaakt. Elke keer als u een 'leuke functie' tegenkomt, kunt u meteen de daarbij horende draaisymmetrische figuur maken.

Tot slot volgt nog een programma voor het tekenen van een fraaie viertoppige grafiek. U bent hem misschien wel eens in een computertijdschrift tegengekomen. Om de snelheid op te voeren is de subroutine 1000 in het hoofdprogramma opgenomen. U kunt dit natuurlijk ook in alle andere programma's doen.

```
 100 REM programma 30
         mooie functie
 110 CLS
 120 INPUT "alpha in graden (45-
135)  ";w
 130 INPUT "verkleiningsfactor (
.3-.55)  ";k
 140 LET u=128: LET v=88
 150 LET h=0.5: LET rd=PI/180
 160 LET c=k*COS (w*rd): LET s=k
*SIN (w*rd)
 170 LET dx=3: LET dy=5: LET k1=
16
 180 DIM h(256)
 190 FOR l=1 TO 256
 200  LET h(l)=-1000
 210 NEXT l
 220 CLS
 230 FOR y=-93 TO 93 STEP dy
 240  LET m1=COS (y*2*PI/93-PI)+
1
 250  FOR x=-93 TO 93 STEP dx
 260   LET m2=COS (x*2*PI/93-PI)
+1
 270   LET z=k1*m1*m2
 280   LET xg=INT (u+x+c*y+h)
 290   LET yg=INT (v+s*y+z+h)
 300   IF x>-93 THEN GO TO 340
 310   LET f1=0: LET l=INT (xg/d
x)+1
 320   IF yg>h(l) THEN LET f1=1:
 LET h(l)=yg
 330   LET x1=xg: LET y1=yg: GO
TO 390
 340   LET f2=0: LET l=INT (xg/d
x)+1
 350   IF yg>h(l) THEN LET f2=1:
 LET h(l)=yg
 360   LET x2=xg: LET y2=yg
 370   IF f1*f2=1 THEN PLOT x1,y
1: DRAW x2-x1,y2-y1
 380   LET x1=x2: LET y1=y2: LET
 f1=f2
 390  NEXT x
 400 NEXT y
 410 IF INKEY$="" THEN GO TO 410
 420 STOP
```

w=45°
k=0,5
k1=20



$$z = \left(\cos\left(\frac{2\pi x}{93}-\pi\right)+1\right) \cdot \left(\cos\left(\frac{2\pi y}{93}-\pi\right)+1\right)$$

6 Turtle-graphics en LOGO-simulatie

LOGO is vooral door de turtle-graphics beroemd geworden. Er is geen andere programmeertaal waarmee zo eenvoudig zeer moeilijke grafische figuren gemaakt kunnen worden dan met LOGO. In dit hoofdstuk zullen we vijf LOGO-programma's in BASIC vertalen. Deze BASIC-programma's bevatten wel 25 tot 30 regels, terwijl LOGO hiervoor slechts 3 of 4 regels nodig heeft.

"Eerst LOGO, daarna andere programmeertalen"

is het motto van vele informatici en pedagogen die zich met de invoering van de informatica op scholen bezighouden.

Wanneer men deze stelling wil begrijpen, moet men niet alleen de taal LOGO en haar mogelijkheden, maar ook de omgeving waarin LOGO ontwikkeld werd goed kennen. Een korte historische terugblik lijkt daarom op zijn plaats.

Het schrijven van computerprogramma's is een intellectuele prestatie en een creatief proces. Aan de wijze waarop wij programma's ontwikkelen, herkennen wij direct onze manier van denken. Aan het Massachusetts Institute of Technology (MIT) heeft men al in het begin van de zestiger jaren een speciale programmeertaal ontwikkeld, met behulp waarvan men kunstmatige intelligentie ging bestuderen. Men noemde deze taal LISP, een afkorting van LISt-Processing.

In LISP werden programma's geschreven die een met kunstmatige zintuigen uitgeruste elektromechanische muis in staat stelden een uitweg uit ieder willekeurig doolhof te vinden. In LISP worden programma's ontwikkeld die de computer tot een medespeler met leervermogen maakt. Hoe meer spelletjes de computer tegen een menselijke tegenspeler speelt, des te beter wordt hij. Goede zetten van de tegenstander onthoudt hij en legt hij vast in zijn geheugen, terwijl hij de eigen slechte zetten voor toekomstige spelletjes uit zijn

geheugen wegveegt. Een 'afvalprodukt' van deze studies aan het MIT zijn de moderne schaakprogramma's.

LOGO is een hoog ontwikkeld dialect van de LISP-taal. Seymour Papert, leerling van de bekende onderzoeker Jean Piaget en medewerker aan het MIT, heeft in twaalf jaar tijd LOGO ontwikkeld. Jean Piaget heeft in zijn bekende boek *Hoe kinderen leren* de kinderlijke denkstructuren laten zien. Hieruit blijkt dat tekenen en knutselen tot de eerste creatieve handelingen van kinderen behoren. Wij zullen zien hoe LOGO deze bezigheden ondersteunt.

Amerikanen gaan door voor een volk dat graag experimenteert. Het was hun echter duidelijk, dat jonge kinderen, wij denken dan aan zes- tot dertienjarigen, niet in staat zijn een programmeertaal zoals BASIC, laat staan Pascal, te leren en algorithmen voor numerieke, niet-numerieke of grafische problemen te schrijven. Daarom heeft men een 'kinderlijke' taal gemaakt (waarachter echter een imponerende software schuilgaat), waarmee het kind met gemak tekeningen kan maken.

Wordt het LOGO-systeem ingeschakeld, tegenwoordig meestal een 64K-microcomputer met bijbehorende software, dan verschijnt in het midden van het beeldscherm een kleine driekhoek, waarvan de top naar boven wijst. De kinderen noemen die driehoek 'turtle', het Engelse woord voor schildpad. Met behulp van eenvoudige bevelen kan het kind de schildpad willekeurig over het beeldscherm laten rondlopen. En al naar gelang het wenselijk is laat de turtle wel of geen spoor na.

De volgende LOGO-opdrachten zijn beschikbaar:

FORWARD	100	= 100 passen vooruit
BACK	50	= 50 passen achteruit
RIGHT	90	= draaiing van 90° met de klok mee
LEFT	45	= draaiing van 45° tegen de klok in
PENUP		= haal pen van papier
PENDOWN		= zet pen op papier
HIDETURTLE		= haal Turtle van het beeldscherm

Normaal gesproken geldt de toestand "PENDOWN". Als we bijvoorbeeld de hoofdletter F op het scherm willen tekenen, kan dat in LOGO als volgt:

```
FORWARD 100  RIGHT 90  FORWARD 50
RIGHT 90  PENUP  FORWARD 50  PENDOWN
RIGHT 90  FORWARD 50
HIDETURTLE
```

Voor 100, 50, 90 en 45 kunnen ook andere waarden gekozen worden.

Omdat LOGO een vertolkend systeem is, wordt elke opdracht (na het geven van RETURN) direct uitgevoerd. Zo op het oog lijkt het slechts leuk speelgoed! Laten we nu eens een LOGO-programma bekijken; liever spreken we van een procedure:

```
TO VIERKANT
REPEAT 4 (FORWARD 75  RIGHT 90)
END
```

Het LOGO-systeem weet dat tussen TO en END een procedure (een stuk programma) gedefinieerd wordt. De procedure tussen TO en END (in ons voorbeeld VIERKANT genoemd) maken we zelf. Zouden we bovenstaande procedure intoetsen, dan zal op het beeldscherm een vierkant met zijden ter lengte 75 getekend worden. De programmeurs onder u kennen vast de opdracht REPEAT waarmee een herhalingsstructuur geprogrammeerd kan worden.

Als we de procedure VIERKANT in het LOGO-systeem ingevoerd hebben, kunnen we hier ook gebruik van maken. LOGO onthoudt alle procedures die wij zelf invoeren. We kunnen dan ook later nieuwe procedures ontwikkelen waarin we gebruik maken van eerder ingevoerde procedures (zoals VIERKANT). Nu volgt een procedure met een parameter:

```
TO VIERKANT:ZIJDE
REPEAT 4 (FORWARD:ZIJDE RIGHT90)
END
```

Toetsen we nu VIERKANT 150 <RETURN> in, dan zal LOGO een vierkant met zijden van 150 tekenen. Niets verhindert u een procedure in te bedden in een volgende procedure en die weer in een volgende en die weer ....., enzovoorts. Alleen de beschikbare geheugenruimte is hierbij een remmende factor. Bekijk het volgende LOGO-programma.

LOGO-programma nr.1

```
TO VIERKANTPATROON
REPEAT 8 (FORWARD 20  LEFT 45  VIERKANT 75)
END
```

Dit eenvoudige drieregelige LOGO-programma tekent een patroon van acht vierkanten (zie p.87). We zullen dit programma straks in BASIC vertalen. Het zal blijken dat hiervoor enige wiskundige en programmeertechnische kennis nodig is. Het LOGO-programma kan door een leerling van de lagere school gemaakt worden, terwijl het

BASIC-programma dat deze acht vierkanten tekent slechts door scholieren van de hoogste klassen van het voortgezet onderwijs kan worden gemaakt. Het wordt nog moeilijker als we intikken:

```
TO STROOK:AANTAL
REPEAT:AANTAL (FORWARD 100  VIERKANTPATROON)
END
```

Tikken we vervolgens in STROOK 5 <RETURN> dan maken we heel eenvoudig een fraai ornament. Probeer dit maar eens in BASIC of Pascal. LOGO wordt pas echt interessant als we van de mogelijkheid gebruik maken dat een procedure zichzelf aanroept (recursiviteit). De turtle-graphics berusten op het principe dat we een procedure parameters geven en dat deze procedure zichzelf steeds met andere parameterwaarden aanroept. Bekijk de volgende procedure:

```
TO VEELHOEK:ZIJDE:HOEK
FORWARD:ZIJDE  LEFT:HOEK
VEELHOEK:ZIJDE:HOEK   ←  hier roept de procedure
END                      zichzelf aan
```

Als we nu VEELHOEK 200 144 intikken, zal het LOGO-systeem een regelmatige vijfpuntige ster tekenen. We zullen echter op de STOP-toets moeten drukken om het tekenen te stoppen. Brengen we in deze procedure een stopmechanisme aan en laten we de procedure steeds de waarde van ZIJDE veranderen, dan krijgen we de welhaast bekendste LOGO-procedure:

LOGO-programma nr. 2

```
TO TURTLE:ZIJDE:GROEI:HOEK
FORWARD:ZIJDE  LEFT:HOEK
MAKE:ZIJDE:ZIJDE+:GROEI
IF:ZIJDE>200 THEN STOP
TURTLE:ZIJDE:GROEI:HOEK
END
```

Als er een dubbele-punt voor een LOGO-woord staat, weet het LOGO-ssyteem dat het om een naam van een variabele gaat en niet om de naam van een procedure of een LOGO-opdracht. In de bovenstaande LOGO-procedure zien we hoe de procedure zichzelf steeds met een nieuwe waarde voor ZIJDE aanroept. Ook dit programma vertalen we in BASIC. Hiermee kunt u elke denkbare rechtlijnige turtle-grafiek maken. U hoeft alleen de waarden van de invoerparameters (ZIJDE, GROEI en HOEK) te veranderen.

Het derde LOGO-programma tekent een reeks vierkanten in spiraalvorm. Het zijn bekende figuren in het 'land van de computergraphics'.

LOGO-programma nr. 3

```
TO VIERKANTSPIRAAL:ZIJDE:HOEK
IF:ZIJDE>150 THEN STOP
VIERKANT:ZIJDE  LEFT:HOEK
VIERKANTSPIRAAL:ZIJDE+4:HOEK
END
```

Ook dit LOGO-programma vertalen we in BASIC. Met deze drie programma's hebt u niet alleen met LOGO maar ook met de 'turtlegraphics' kennis gemaakt. Wilt u meer weten van deze graphicswereld, dan wijzen wij op het boek *Turtle Geometry: The Computer as a Medium for Exploring Mathematics* van Harold Abelson en Andrea di Sessa, uitgegeven door Cambridge, MA:MIT Press, 1981. Liefhebbers van wiskunde, informatica en LOGO vinden in dit boek een schat aan informatie.

Natuurlijk zijn we in LOGO niet gebonden aan 'rechte lijnen'. In LOGO hebben we de beschikking over alle wiskundige functies en bewerkingen. Alle programma's uit dit boek kunnen in LOGO geschreven worden. Zij zouden dan vast korter, overzichtelijker en begrijpelijker geweest zijn. In LOGO kunnen we heel eenvoudig met lijsten en tabellen werken. In LOGO hoeven we de variabelen niet te declareren. Achter een naam in LOGO kan gewoon een getal, een string, een n-dimensionaal veld en nog veel meer schuil gaan. Ook kan het de naam van een procedure zijn. We hoeven ons niet te bekommeren om het reserveren van geheugenruimte (DIM-opdracht in BASIC), noch om het aangeven van het type (real, integer, character, boolean, enz.); het LOGO-systeem regelt dit zelf.

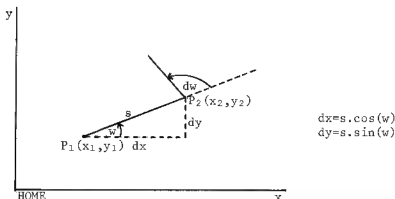
Wie in LOGO programmeert leert automatisch netjes programmeren. Begrippen als procedure, recursie, herhaling, toekenning worden op een natuurlijke manier aangeleerd. Met een paar opdrachten is een kind al in staat ingewikkelde figuren te tekenen. Met LOGO degraderen we de computer niet tot een programmeerbare zakrekenmachine, maar halen we er alles uit wat erin zit.

Vertaling LOGO-programma nr.1 in BASIC

De kern van elk LOGO-graphics-programma wordt gevormd door twee opdrachten:

```
FORWARD (resp. BACK)   :ZIJDE
LEFT (resp. RIGHT)     :HOEK
```

Hoe vertalen we deze opdrachten in BASIC? Bekijk hiertoe de volgende figuur.

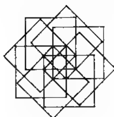


De PEN staat in punt $P_1(x_1,y_1)$ en moet in de richting van hoek w over een afstand s verplaatst worden. In $P_2(x_2,y_2)$ aangekomen moet de PEN (eigenlijk de turtle) een hoek van dw° linksom maken. De volgende BASIC-opdrachten voeren dit uit:

```
INPUT x1,y1,w,s,dw
LET h=0.5: LET rd=PI/180: LET w1=w*rd
LET x2=(x1+s*COS(w1)+h)
LET y2=(y1+s*SIN(w1)+h)
PLOT x1,y1: DRAW x2-x1,y2-y1
LET x1=x2: LET y1=y2
LET w=w+dw: IF w>360 THEN LET w=w-360
LET w1=w*rd
```

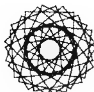
Deze opdrachten komt u in alle volgende BASIC-programma's tegen. Meer theorie is niet nodig! De programma's moeten met bovenstaande informatie gelezen kunnen worden.

Experimenteer met programma 31 LOGO-1 vierkantpatroon. Met w=90°, s1=20, dw=45, s2=75 en n=8 krijgen we de onderstaande tekening.



```
 100 REM programma 31
      vierkantspatronen (LOGO-1)
 110 INPUT "coordinaten startpun
t    ";x1,y1
 120 INPUT "beginrichting (grade
n)   ";w
 130 INPUT "verplaatsing    ";s1
 140 INPUT "draaihoek ,linksom
";dw
 150 INPUT "zijde vierkant    ";s
2
 160 INPUT "aantal vierkanten  "
;n
 170 LET h=0.5: LET rd=PI/180: L
ET w1=w*rd
 180 CLS
 190 FOR j=1 TO n
 200  LET x2=INT (x1+s1*COS (w1)
+h)
 210  LET y2=INT (y1+s1*SIN (w1)
+h)
 220  PLOT x1,y1: DRAW x2-x1,y2-
y1
 230  LET x1=x2: LET y1=y2
 240  LET ax=x1: LET ay=y1
 250  LET w=w+dw: IF w>360 THEN
LET w=w-360
 260  LET w1=w*rd
 270  FOR k=0 TO 2
 280   LET x2=INT (x1+s2*COS (w1
+k/2*PI)+h)
 290   LET y2=INT (y1+s2*SIN (w1
+k/2*PI)+h)
 300  PLOT x1,y1: DRAW x2-x1,y2-
y1
 310   LET x1=x2: LET y1=y2
 320  NEXT k
 330  PLOT x1,y1: DRAW ax-x1,ay-
y1
 340  LET x1=ax: LET y1=ay
 350 NEXT j
 360 IF INKEY$="" THEN GO TO 360
 370 STOP
```

w=90, s1=30, dw=48, s2=70 en n=15 geeft de onderstaande tekening. Als dw een deler is van 360, is n gelijk aan 360/dw. Kiest u voor dw een willekeurige waarde, neem dan voor n een groot getal, bijvoorbeeld 100, en breek het tekenen met de stoptoets af. De 'kinderen' doen dat in LOGO ook op deze manier.



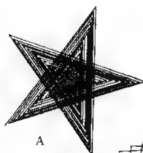
Vertaling LOGO-programma nr.2 in BASIC

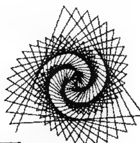
Het programma lijkt sterk op het vorige en behoeft daarom geen commentaar.

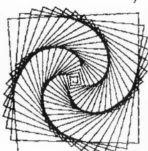
```
 100 REM programma 32
     turtle grafiek (LOGO-2)
 110 INPUT "coordinaten startpun
t    ";x1,y1
 120 INPUT "beginrichting (grade
n)  ";w
 130 INPUT "verplaatsing    ";s
 140 INPUT "draaihoek ,linksom
";dw
 150 INPUT "toename zijde  ";ds
 160 LET h=0.5: LET rd=PI/180: L
ET w1=w*rd
 170 CLS
 180 LET x2=INT (x1+s*COS (w1)+h
)
 190 LET y2=INT (y1+s*SIN (w1)+h
)
 200 IF x2<0 OR x2>255 OR y2<0 O
R y2>175 THEN GO TO 260
 210  PLOT x1,y1: DRAW x2-x1,y2-
y1
 220  LET x1=x2: LET y1=y2
 230  LET w=w+dw: IF w>360 THEN
LET w=w-360
 240  LET w1=w*rd: LET s=s+ds
 250 GO TO 180
 260 IF INKEY$="" THEN GO TO 260
 270 STOP
```

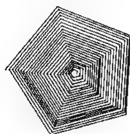
Het is handig de ingevoerde waarden na het tekenen op het scherm of op een printer te laten afdrukken. In de volgende tabel zien we de diverse waarden voor de daaronderstaande tekeningen.

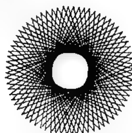
Tekening	x1	y1	w	s	dw	ds
A	128	88	90	5	144	3
B	128	88	90	5	123	2
C	128	88	90	-2	92	2
D	128	88	90	5	72	1
E	128	140	90	130	145	0

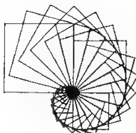

A


B


C


D


E



Vertaling LOGO-programma nr.3 in BASIC

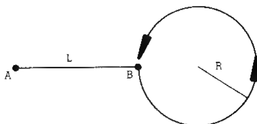
```
 100 REM programma 33
        vierkantspiraal (LOGO-3)
 110 INPUT "coordinaten startpun
t    ";x1,y1
 120 INPUT "beginrichting (grade
n)   ";w
 130 INPUT "zijde beginvierkant
  ";s
 140 INPUT "draaihoek ,linksom
";dw
 150 INPUT "toename zijde  ";ds
 160 LET h=0.5: LET rd=PI/180: L
ET w1=w*rd: LET vlag=0
 170 CLS
 180 LET ax=x1: LET ay=y1
 190 FOR f=0 TO PI STEP PI/2
 200  GO SUB 1000
 210  IF vlag=1 THEN GO TO 300
 220  PLOT x1,y1: DRAW x2-x1,y2-
y1
 230  LET x1=x2: LET y1=y2
 240 NEXT f
 250 PLOT x1,y1: DRAW ax-x1,ay-y
1
 260 LET x1=ax: LET y1=ay
 270 LET w=w+dw: IF w>=360 THEN
LET w=w-360
 280 LET w1=w*rd: LET s=s+ds
 290 GO TO 190
 300 IF INKEY$="" THEN GO TO 300
 310 STOP
1000 LET x2=INT (x1+s*COS (w1+f)
+h)
1010 LET y2=INT (y1+s*SIN (w1+f)
+h)
1020 IF x2<0 OR x2>255 OR y2<0 O
█ y2>175 THEN LET vlag=1
1030 RETURN
```

Voor de boven het programma staande tekening kozen we x1=100, y1=120, w=90, s=5, dw=15 en ds=3,5.

Vergelijk deze drie BASIC-programma's eens met de overeenkomstige LOGO-programma's. In BASIC moeten we zelf alle graphic-software schrijven, terwijl deze in LOGO als machine-routines aanwezig is.

Het 4e en 5e LOGO-programma

Tot slot van dit hoofdstuk geven we nog twee BASIC-programma's waarmee de turtle behalve rechte wegen ook kromme wegen kan bewandelen. Dit voegt een nieuw element aan de turtle-graphics toe. De LOGO-structuur zullen we stapsgewijs verfijnen. Als uitgangspunt nemen we de onderstaande figuur. Deze bestaat uit een lijnstuk L, met daaraan vast een cirkel met straal R. We noemen deze figuur voor het gemak even 'de steelpan'.



Als de turtle de steelpan tekent, begint hij in A. Hij legt vervolgens een afstand L in positieve x-richting af en komt in B. Hier draait hij 90° met de klok mee en legt daarna de omtrek van de cirkel af tot hij weer in B uitkomt. Daar draait hij zich 90° met de klok mee en kijkt dan rechtvooruit naar het punt A. Met deze steelpan kunnen we twee soorten tekeningen maken. Laten we deze mogelijkheden bekijken:

LOGO-programma 4

```
TO CIRKELFIGUUR-1:LIJNSTUK:STRAAL:HOEK
STEELPAN:LIJNSTUK:STRAAL
LEFT:HOEK
CIRKELFIGUUR-1:LIJNSTUK:STRAAL:HOEK
END
```

LOGO-programma 5

```
TO CIRKELFIGUUR-2:LIJNSTUK:STRAAL:HOEK
REPEAT 4 (STEELPAN:LIJNSTUK:STRAAL  LEFT 90)
LEFT:HOEK
CIRKELFIGUUR-2:LIJNSTUK:STRAAL:HOEK
END
```

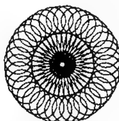
Deze twee LOGO-programma's kunnen nog niet uitgevoerd worden, want de procedure STEELPAN is nog niet in het LOGO-systeem gedefinieerd. We gaan de structuur verfijnen:

```
TO STEELPAN:LIJNSTUK:STRAAL
FORWARD:LIJNSTUK  RIGHT 90
CIRKEL:STRAAL
RIGHT 90
END
```

Ook in deze procedure zien we een nog niet gedefinieerde procedure, CIRKEL, opduiken. Om de turtle een cirkel te kunnen laten maken zullen we deze procedure CIRKEL moeten maken. Veel LOGO-versies bevatten reeds zo'n voorgedefinieerde CIRKEL-procedure.

Wij gebruiken in het navolgende programma de CIRCLE-functie van de ZX Spectrum. In de regels 230 t/m 260 zien we dan ook:

```
230 LET ax=x2: LET ay=y2
240 LET u=INT(ax+r*COS(w1)+h)
250 LET v=INT(ay+r*SIN(w1)+h)
260 CIRCLE u,v,r
```



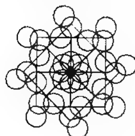
Vertaling LOGO-programma nr.4 in BASIC

```
 100 REM programma 34
          cirkelfiguur-1 (LOGO-4)
 110 CLS
 120 INPUT "coordinaten startpun
t  ";x1,y1
 130 INPUT "beginrichting (grade
n)  ";w
 140 INPUT "lengte van de steel
 ";l
 150 INPUT "straal van de cirkel
  ";r
 160 INPUT "draaihoek , linksom
 ";dw
 170 LET h=0.5: LET rd=PI/180: L
ET w1=w*rd
 180 CLS
 190 LET xx=x1: LET yy=y1
 200 LET x2=INT (x1+l*COS (w1)+h
)
 210 LET y2=INT (y1+l*SIN (w1)+h
)
 220 PLOT x1,y1: DRAW x2-x1,y2-y
1
 230 LET ax=x2: LET ay=y2
 240 LET u=INT (ax+r*COS (w1)+h)
 250 LET v=INT (ay+r*SIN (w1)+h)
 260 CIRCLE u,v,r
 270 LET w=w+dw: IF w>=360 THEN
LET w=w-360
 280 LET w1=w*rd: LET x1=ax: LET
 y1=ay
 290 IF x1=xx AND y1=yy THEN GO
TO [illegible]0
 300 GO TO 200
 310 IF INKEY$="" THEN GO TO 310
 320 STOP
```

De bovenstaande tekening krijgen we door x1=128, y1=120, w=90, l=80, r=20 en dw=170 te kiezen.

Vertaling LOGO-programma nr.5 in BASIC

```
 100 REM programma 35
          cirkelfiguur-■ (LOGO-5)
 110 CLS
 120 INPUT "coordinaten startpun
t  ";x1,y1
 130 INPUT "beginrichting (grade
n)  ";w
 140 INPUT "lengte van de steel
 ";l
 150 INPUT "straal van de cirkel
  ";r
 160 INPUT "draaihoek , linksom
 ";dw
 170 LET h=0.5: LET rd=PI/180: L
ET w1=w*rd
 180 CLS
 190 FOR j=1 TO 4
 200  LET x2=INT (x1+l*COS (w1)+
h)
 210  LET y2=INT (y1+l*SIN (w1)+
h)
 220  PLOT x1,y1: DRAW x2-x1,y2-
y1
 230  LET ax=x2: LET ay=y2
 240  LET u=INT (ax+r*COS (w1)+h
)
 250  LET v=INT (ay+r*SIN (w1)+h
)
 260  CIRCLE u,v,r
 270  LET w=w+90: IF w>360 THEN
LET w=w-360
 280  LET w1=w*rd: LET x1=ax: LE
T y1=ay
 290 NEXT j
 300 LET w=w+dw: IF w>=360 THEN
LET w=w-360
 310 LET w1=w*rd
 320 GO TO 190
 330 STOP
```



Voor deze tekening geldt: x1=128, y1=88, w=0, l=40, r=10 en dw=45. Met andere waarden voor deze variabelen kunt u de mooiste turtle-plaatjes maken.

7 *Educatieve toepassings-programma's*

De voorgaande 35 grafische programma's zijn geen toepassingsprogramma's. Ze waren bedoeld om een overzicht te geven van de mogelijkheden van graphics met hoog oplossend vermogen. In dit hoofdstuk zullen we vijf praktijkgerichte grafische programma's presenteren. Deze programma's zijn:

1. Tekenen van een landkaart
2. Maken van een histogram (een stavengrafiek)
3. Demonstratieprogramma voor de breking van lichtstralen
4. Demonstratieprogramma voor de 'speldenworp' van Buffon
5. Prooi-roofdierpopulaties

1. Tekenen van een landkaart

Educatieve programma's over topografie gebruiken dikwijls landkaarten van een land of van een werelddeel die door de computer getekend worden. We zullen laten zien hoe de computer zo'n landkaart kan tekenen. We gaan de landkaart (althans de grensomtrek) van Zwitserland tekenen. Waarom Zwitserland? Wel, in de eerste plaats omdat de auteur dit gekozen heeft en in de tweede plaats omdat Zwitserland geen eilanden heeft. Het tekenen van de contouren van Nederland gaat in principe net zo, alleen kosten al die eilanden een hoop extra werk, vandaar!

Hoe tekenen we nu de omtrek van een bepaald land? Heel eenvoudig! We pakken gewoon een atlas, een stukje overtrekpapier, een potlood, een lineaal en millimeterpapier. We kiezen in de atlas een land of werelddeel uit en trekken het op overtrekpapier over. Daarna leggen we het overgetrokken plaatje op millimeterpapier, waar we van te voren een coördinatenstelsel op getekend hebben (een x- en een y-as). We bepalen nu van een (groot) aantal punten op de omtrek van het land (of werelddeel) de coördinaten (x,y) en schrijven deze op. Als we ons op het beeldscherm ook een coördinatenstelsel voorstellen en als we hierin de punten, waarvan we de coör-

dinaten in een programma opnemen, met elkaar laten verbinden, dan ontstaat een 'landkaart' op het scherm.

In het onderstaande programma, dat 'de kaart' van Zwitserland tekent, is een 'echte' kaart gebruikt met een schaal van 1 : 2.000.000. Om een enigszins natuurgetrouwe weergave te verkrijgen zijn de coördinaten van 90 grenspunten berekend. De coördinaten van deze punten, die opgegeven zijn in millimeters ten opzichte van de oorsprong van het gekozen coördinatenstelsel, zijn in DATA-regels opgenomen. Als het programma het coördinatenpaar (0,0) leest, weet het dat de 'kaart' af is.

De x-coördinaten liggen tussen 6 en 177. Om deze naar het hoge-resolutiebereik 0-255 te transformeren vermenigvuldigen wij zowel de x- als de y-coördinaten met de factor k = 1,3. Hierdoor blijft nog wat ruimte over om een kader rond de kaart te tekenen. De werking van het programma zal hiermee duidelijk zijn.

We hebben op deze manier ook een kaart van Europa en zelfs een wereldkaart getekend. Het probleem met de eilanden hebben we als volgt opgelost. Als het programma in een DATA-regel negatieve x- en y-coördinaten leest, weet het programma dat dit punt niet met het vorige verbonden moet worden en dat dit punt dus het begin is van een apart stukje 'land'.



```
 100 REM programma 36
         kaart van Zwitserland
 110 CLS
 120 LET k=1.3: LET h=0.5: LET u
=15: LET v=175
 130 READ x,y
 140 LET x1=INT (u+k*x+h): LET y
1=INT (v-k*y+h)
 150 LET x2=INT (u+k*x+h): LET y
2=INT (v-k*y+h)
 160 PLOT x1,175-y1: DRAW x2-x1,
y1-y2
 170 LET x1=x2: LET y1=y2
 180 READ x,y
 190 IF x<>0 THEN GO TO 150
 200 IF INKEY$="" THEN GO TO 200
 210 STOP
 220:
 300 DATA 69,108,71,107,70,104,7
5,104,75,104,76,106
 310 DATA 80,107,81,104,86,105,9
1,108,94,107
 320 DATA 101,106,100,106,105,10
8,106,110,101,110
 330 DATA 98,112,102,117,108,118
,112,112,114,115
 340 DATA 118,110,126,110,139,10
2,145,103,148,95
 350 DATA 142,86,144,78,154,77,1
54,77,154,72,163,67
 360 DATA 158,58,173,76,177,73,1
74,59,177,56
 370 DATA 177,52,171,51,167,56,1
61,50,165,43
 380 DATA 166,34,162,34,157,42,1
43,38,139,46
 390 DATA 136,45,133,48,132,40,1
22,23,125,15
 400 DATA 122,11,119,12,114,20,1
16,25,108,35
 410 DATA 102,45,94,42,88,35,90,
26,79,15
 420 DATA 75,15,66,19,60,14,52,1
2,48,13
 430 DATA 37,29,39,36,37,40,39,4
3,29,45
 440 DATA 16,38,16,33,13,29,6,28
,6,32
 450 DATA 11,34,13,40,10,45,12,5
3,28,63
 460 DATA 26,73,30,73,48,94,42,9
4,46,102
 470 DATA 54,102,53,99,61,98,63,
102,69,108
 480 DATA 0,0
```

In dit programma leggen we de oorsprong (u,v) linksbovenaan het scherm; u=15 en v=175. De coördinaatparen x,y zijn namelijk door de auteur ten opzichte van een oorsprong in de linkerbovenhoek berekend.
Kijk in de appendix voor een uitbreiding van dit programma.

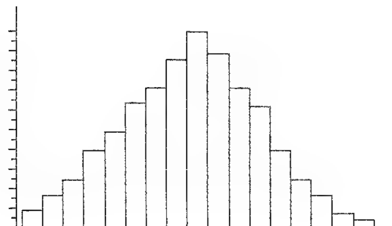
2. Maken van een histogram

Vaak willen we een aantal waarnemingen grafisch in een stavendiagram of histogram weergeven. Hiertoe dient het volgende programma. Dit programma tekent een horizontale en een verticale as. De verticale as wordt in stukjes van acht schermpuntjes verdeeld. Dit komt overeen met 5% van de hele verticale as. Elk tweede streepje op deze as geeft de volgende 10% aan en is iets breder getekend. Hiermee kan de lengte van de staven redelijk geschat worden.

Het programma kan maximaal 100 waarnemingen verwerken. Uit estetische overwegingen moet u echter niet veel meer dan 40 waarnemingen invoeren, omdat de staafjes anders meer op lijntjes dan op balkjes gaan lijken.

De waarnemingen worden zo 'geschaald' dat de grootste waarneming overeenkomt met de lengte van de verticale as.

Het zou mooi zijn als we bij de staven, de assen en de schaalverdeling tekst en getallen konden zetten, maar dat gaat heel lastig, dus doen we het nu maar niet. Denkt u erom dat de getallen die u intoetst de lengte van de staven aangeven. Als elke staaf een bepaalde klasse met waarnemingen voorstelt, dan voert u dus steeds het aantal waarnemingen (de frequentie) van een klasse in. Het programma kan uitgebreid worden door er een stuk voor te zetten dat ruwe gegevens inleest, die vervolgens netjes in klassen verdeeld worden. De frequentie van de waarnemingen in de klassen wordt vervolgens gebruikt om de stavengrafiek te tekenen.

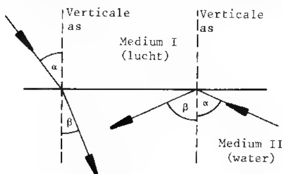


```
 100 REM programma 37 histogram
 110 CLS
 120 PRINT "HISTOGRAM TEKENEN"
 130 PRINT "-----------------"
 140 PRINT : PRINT
 150 INPUT "Hoeveel gegevens (<4
0)  ";n
 160 DIM a(n)
 170 LET mx=-1e10: PRINT
 180 FOR j=1 TO n
 190  PRINT "Waarde ";j;TAB (12)
;
 200  INPUT a(j)
 205  PRINT a(j)
 210  IF a(j)>mx THEN LET mx=a(j
)
 220 NEXT j
 230 CLS
 240 REM horizontale as
 250 PLOT 0,10: DRAW 255,0
 260 REM verticale as
 270 PLOT 10,10: DRAW 0,165
 280 REM schaalverdeling
 290 FOR j=1 TO 10
 300  LET x1=4: LET y1=j*16
 310  LET x2=10: LET y2=y1
 320  PLOT x1,y1: DRAW x2-x1,y2-
y1
 330 LET x1=7: LET y1=y2+8: LET
y2=y1
 340  PLOT x1,y1: DRAW x2-x1,y2-
y1
 350 NEXT j
 360 REM staven tekenen b=breedt
e
 370 LET b=INT (235/n): LET h=0.
5
 380 FOR j=1 TO n
 390  LET x1=(j-1)*b+15: LET y1=
10
 400  LET x2=x1: LET y2=INT (10+
165*a(j)/mx+h)
 410  PLOT x1,y1: DRAW x2-x1,y2-
y1
 420  LET x1=x2: LET y1=y2: LET
x2=x1+b: LET y2=y1
 430  DRAW x2-x1,y2-y1
 440  LET x1=x2: LET y1=y2: LET
y2=10
 450  DRAW x2-x1,y2-y1
 460 NEXT j
 470 IF INKEY$="" THEN GO TO 470
 480 STOP
```

3. Demonstratieprogramma voor de breking van lichtstralen

Met dit programma willen we laten zien hoe we de ZX Spectrum bij natuurkundelessen zouden kunnen gebruiken. Voordat we het programma geven leggen we nog iets uit van de natuurkundige beginselen van de breking van licht.

Als een lichtstraal vanuit de ene stof (bijvoorbeeld lucht) een andere, optisch dichtere, stof (bijvoorbeeld water) binnenkomt, wordt de straal op het scheidingsvlak van beide stoffen naar de verticale as toe gebogen. Zie onderstaande figuur.



Hierbij geldt de brekingswet van Snellius:

$$\frac{\sin\alpha}{\sin\beta} = \frac{c_1}{c_2} = n$$

c_1 en c_2 zijn de lichtsnelheden in respectievelijk de eerste en de tweede stof. De constante n heet de brekingsindex. Bij de overgang van lucht naar water geldt een brekingsindex van 1,33. Voor de overgang van lucht naar glas gelden andere waarden, enzovoorts.

Als het licht vanuit een 'dichter' medium overgaat in een 'dunner' medium geldt het omgekeerde: de lichtstralen worden nu van de verticale as, die loodrecht op het scheidingsvlak van beide stoffen staat, afgebogen. De brekingshoek α kan nooit groter dan 90° zijn.

Voor dit grensgeval (α = 90°) geldt:

$$\frac{\sin 90°}{\sin\beta^*} = n \Rightarrow \sin\beta^* = \frac{1}{n} \quad \text{(want } \sin 90° = 1\text{)}$$

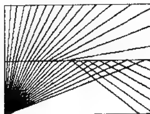
Bij de overgang van lucht naar water geldt voor β^*:

$$\sin\beta^* = \frac{1}{1,33} \Rightarrow \sin\beta^* = 0,7519 \Rightarrow \beta^* = 48,75°$$

Groter dan 48,75° kan β dus niet worden. Dit betekent dat als het licht van water overgaat in lucht met een invalshoek α die groter is dan β^* (zie rechter figuur), het licht niet meer het water 'uitkomt'. Op het scheidingsvlak van water en lucht wordt het licht geheel teruggekaatst. Op dit principe berusten de moderne glasvezelkabels, waarin informatie in de vorm van licht wordt getransporteerd.

In het onderstaande demonstratieprogramma kan de brekingsindex n ingetoetst worden. De lichtbron bevindt zich in het punt met schermcoördinaten (0,10). Het scheidingsvlak ligt horizontaal en is de lijn y=88. De invalshoek van een lichtstraal die van medium 2 (de dichtere stof, onderste helft) in medium 1 (de lichtere stof, bovenste helft) overgaat wordt van 0° steeds met stapjes van 3° opgehoogd. Op het moment dat deze invalshoek groter wordt dan β^* (deze is afhankelijk van de ingetoetste brekingsindex) treedt totale reflectie (terugkaatsing) op. Rond de figuur wordt een kader getekend. De eindpunten van de diverse lichtstralen worden met trigonometrische functies berekend. De commentaaropdrachten in het programma leggen nog eens uit 'wat waar' gebeurt.

De inverse functie van de sinusfunctie (de boogsinus of arcsinus) bestaat niet op de ZX Spectrum. We lossen dit op door de inverse tangensfunctie (ATN in BASIC) te gebruiken.

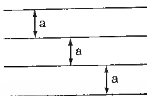


Het voordeel van een dergelijke computersimulatie, boven het uitvoeren van het natuurkundige experiment, is dat we heel gemakkelijk verschillende brekingsindexen kunnen invoeren zonder steeds andere apparatuur op te hoeven stellen en dat de lichtstralen als dunne lijnen zichtbaar gemaakt kunnen worden.

```
 100 REM programa 38
          breking van licht
 110 CLS
 120 INPUT "brekingsindex n ";n
 130 LET v=88: LET h=0.5: LET rd
=PI/180: LET b=0
 140 PLOT 0,v: DRAW 255,0
 150 PLOT 0,10: DRAW 255,0
 160 DRAW 0,165: DRAW -255,0: DR
AW 0,-165
 170 REM stralen in medium 1 en
2 tekenen
 180 REM b=beta in graden; b1=be
ta in radialen
 190 REM a1=alpha in radialen; s
a=SIN (a1)
 200 LET b=b+3: LET b1=b*rd: LET
 x1=0: LET y1=10
 210 LET x2=INT (77*TAN (b1)+h):
 LET y2=v
 220 IF x2>255 THEN GO TO 370
 230 PLOT x1,y1: DRAW x2-x1,y2-y
1: LET x1=x2: LET y1=y2
 240 REM SIN (alpha) en alpha be
rekenen
 250 REM controle op totale refl
ectie
 260 LET s=n*SIN (b1): IF s>1 TH
EN GO TO 330
 270 LET a1=ATN (s/SQR (1-s*s))
 280 LET x2=INT (x1+77*TAN (a1)+
h): LET y2=165
 290 IF x2<255 THEN PLOT x1,y1:
DRAW x2-x1,y2-y1: GO TO 200
 300 LET x2=255: LET y2=INT (v+(
255-x1)/TAN (a1)+h)
 310 PLOT x1,y1: DRAW x2-x1,y2-y
1: GO TO 200
 320 REM totale terugkaatsing
 330 LET x2=x1+x1: LET y2=10
 340 IF x2<255 THEN PLOT x1,y1:
DRAW x2-x1,y2-y1: GO TO 200
 350 LET x2=255: LET y2=INT (v-(
255-x1)/TAN (b1)+h)
 360 PLOT x1,y1: DRAW x2-x1,y2-y
1: GO TO 200
 370 IF INKEY$="" THEN GO TO 370
 380 STOP
```

4. De speldenworp van Buffon (1773)

Stelt u zich voor dat we in een plat vlak een aantal evenwijdige lijnen met een onderlinge afstand a tekenen.



We werpen nu, zonder echt te 'mikken', een speld met een lengte c die kleiner dan of gelijk aan a is ($c \leq a$) op het vlak met de evenwijdige lijnen. Hoe groot is nu de kans dat een speld een van de lijnen treft, dat wil zeggen snijdt of raakt?



speld snijdt een lijn



speld raakt een lijn

Dit probleem kwam in 1773 op bij de Franse wiskundige Buffon na het zien van de Amerikaanse vlag met de 'stars en stripes'. Buffon heeft berekend dat deze kans gelijk is aan

$$\frac{2c}{a\pi}$$

Omdat deze formule de constante π bevat, heeft men deze 'speldenworp' vaak gebruikt om de waarde van π door simulatie te bepalen. We werpen hiertoe vaak, bijvoorbeeld een lucifer, op een papier waarop een aantal evenwijdige lijnen (met een onderlinge afstand die groter dan of gelijk aan de lengte van de lucifer is). Als de lucifer in n worpen k keer een lijn treft, dan geldt dat

$$\frac{2c}{a\pi} \quad \text{ongeveer gelijk is aan } \frac{k}{n}.$$

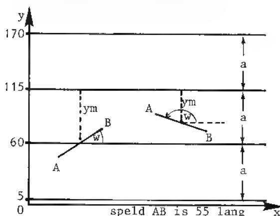
Beide uitdrukkingen geven namelijk een indruk van de kans dat de lucifer een lijn treft.
We kunnen als volgt de waarde van π benaderen:

$$\frac{2c}{a\pi} = \frac{k}{n} \Rightarrow \pi \simeq \frac{2cn}{ak}.$$

Op deze manier heeft de astronoom Rudolf Wolf in 1850 met 5000 luciferworpen voor π de waarde 3,1596 gevonden (π = 3,141592654....).

Met een computer kunnen we gemakkelijk duizenden worpen simuleren. Deze programma's vinden we in bijna elk informaticaleerboek. Dit zijn echter programma's die vrij lang draaien, maar waaraan niets te 'zien' is. Als we bijvoorbeeld in zo'n programma voor n de waarde 10.000 zouden intikken, zien we eerst een tijd niets en vervolgens verschijnt de mededeling dat van de 10.000 keer 6.349 keer een lijn is geraakt, hetgeen voor π de waarde 3,1501024 oplevert.

We gaan een programma maken dat ook dergelijke berekeningen uitvoert, maar dat bovendien elke speld die geworpen wordt laat zien. We zien het experiment als een film aan ons voorbijgaan. Bekijk hiertoe de onderstaande tekening.



De manier waarop een speld valt kan met twee toevalsgetallen bepaald worden. De twee toevalsgetallen bepalen respectievelijk de afstand ym van de speld tot de daarboven liggende lijn en de hoek w die de speld met de lijnen maakt:

$0 \leq ym \leq a$	;	ym is de afstand van het midden van de speld tot de daarboven liggende horizontale lijn
$0 \leq w \leq 180°$	;	w is de hoek (in positieve zin, dat wil zeggen bij draaiing tegen de klok in) die de speld met de positieve x-richting maakt

Hierna volgt een illustratie van hoe het eruit kan zien en het programma.

```
100 REM programma 39
        speldenworp van Buffon
110 CLS
120 PRINT "SPELDENWORP VAN BUFF
ON"
130 PRINT "--------------------
--"
140 INPUT "Hoeveel worpen ";n
150 LET m=0: LET h=0.5
160 CLS
170 REM lijnen trekken
180 FOR y=5 TO 170 STEP 55
190   PLOT 0,y: DRAW 255,0
200 NEXT y
210 REM n maal gooien en tekene
n
220 FOR j=1 TO n
230   LET xm=INT (195*RND+30+h)
240   LET ym=INT (55*RND+60+h)
250   LET w=PI*RND
260   LET dx=27.5*COS (w): LET d
y=27.5*SIN (w)
270   LET x1=INT (xm-dx+h): LET
y1=INT (ym-dy+h)
280   LET x2=INT (xm+dx+h): LET
y2=INT (ym+dy+h)
290   PLOT x1,y1: DRAW x2-x1,y2-
y1
300   IF y1<=60 OR y2>=115 THEN
LET m=m+1
310 NEXT j
320 IF INKEY$="" THEN GO TO 320
330 CLS
340 PRINT "aantal worpen";TAB (
21);n
350 PRINT "aantal keer snijden
";TAB (21);m
360 PRINT "benadering voor pi";
TAB (21);2*n/m
370 STOP
```



Uit de waarden voor ym en w berekenen we de coördinaten van de uiteinden A en B van de speld. Het programma tekent hiermee de speld op het beeldscherm.

Een speld treft een lijn als de y-coördinaat van A groter dan of gelijk aan 115 is of als de y-coördinaat van B kleiner dan of gelijk aan 60 is. De x-coördinaten van A en B doen er in het geheel niet toe.

Het programma tekent als 'speelveld' vier evenwijdige lijnen met een onderlinge afstand van 55. De spelden zijn trouwens ook 55 lang. Als alle spelden geworpen zijn kunnen door een toets in te drukken de waarden voor n, k en π worden afgelezen.

Verander de regels 320 t/m 370 als volgt:

```
320 PRINT AT 18,1;"aantal worpen";TAB(21);n
330 PRINT AT 19,1;"aantal keer snijden";TAB(21);m
340 PRINT AT 20,1;"benadering voor pi";TAB(21);2*n/m
350 IF INKEY$="" THEN GO TO 350
360 STOP
```

Nu worden de gegevens over het aantal worpen en het aantal keer snijden en over de benadering van pi onder de tekening afgedrukt.

5. Prooi-roofdierpopulaties

Het laatste educatieve programma is een (deterministische) simulatie van een ecologisch systeem. Het is een demonstratieprogramma voor een biologieles.

Het ecologische systeem bevat gras, hazen en vossen. Tussen deze drie ecologische componenten gelden de volgende betrekkingen:

1. De hazen eten gras en de vossen eten hazen.
2. Als er meer gras groeit, neemt ook het aantal hazen toe. Deze hazen eten echter van het gras en verminderen zo hun eigen groei.
3. Als er meer hazen komen, neemt ook het aantal vossen toe. Omdat vossen hazen eten, verminderen zij zelf hun groei, net zoals bij de hazen en het gras.

Als we het aantal hazen op een bepaald tijdstip t aangeven met h(t) en het aantal vossen met v(t), kunnen we, volgens Lotka en Volterra (1920), voor het aantal hazen en vossen op tijdstip t+1 de volgende vergelijkingen opstellen:

$$h(t+1) = h(t) + a.h(t) - b.h(t).v(t) \qquad \text{vergelijking (1)}$$
$$v(t+1) = v(t) + c.v(t).h(t) - d*v(t) \qquad \text{vergelijking (2)}$$

De toename van het aantal hazen tussen de tijdstippen t en t+1 is evenredig met het aantal hazen op tijdstip t, dus

$$h(t+1)-h(t) = a.h(t)$$

h(t+1)-h(t): toename hazen
a: groeifactor
h(t): aantal hazen op tijdstip t

De afname van het aantal hazen is evenredig met het aantal aanwezige hazen (natuurlijk verloop) en met het aantal vossen, want die eten hazen, dus

$$\underbrace{h(t+1)-h(t)}_{\text{groei van het aantal hazen}} = \underbrace{a.h(t)}_{\text{toename aantal hazen}} - \underbrace{b.h(t).v(t)}_{\text{afname aantal hazen}}$$

Dit is vergelijking (1). We nemen in het programma aan dat er altijd genoeg gras voor de hazen voorhanden is.

De toename van het aantal vossen is evenredig met het aantal aanwezige vossen en met het aantal hazen (prooi). De afname van het aantal vossen is alleen evenredig met het aantal, omdat in dit systeem de vossen zelf geen prooidieren zijn. Voor de vossen krijgen we dus:

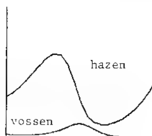
$$\underbrace{v(t+1)-v(t)}_{\text{groei van het aantal vossen}} = \underbrace{c.v(t).h(t)}_{\text{toename aantal vossen}} - \underbrace{d.v(t)}_{\text{afname aantal vossen}}$$

De vergelijkingen zijn als zogeheten differentievergelijkingen opgesteld. Het gras speelt, zoals gezegd, eigenlijk geen rol; er is altijd genoeg om alle hazen te voeden.

Bij het draaien van het simulatieprogramma hebben we de volgende waarden gekozen:
x (= h(0)) = 200; y (= v(0)) = 20; a=0,3; b=0,01; c=0,002 en d=0,5.

De grafiek die het programma tekent laat heel mooi de groei en de afname van de hazen- en vossenpopulaties zien. Het aantal hazen groeit eerst, bereikt een maximum en neemt vervolgens af. De vossen groeien ook, maar later, bereiken later een maximum en nemen dan ook af, waarna de cyclus zich herhaalt. In de biologie noemen we dit een dynamisch evenwicht.

U hoeft de waarden voor de coëfficiënten a, b, c en d maar iets te veranderen of het systeem kan ontregeld worden. Hierbij neemt òf het aantal hazen enorm toe òf alle hazen en vossen sterven snel uit. Deze eenvoudige simulatie toont aan hoe desastreus een kleine ingreep in een bestaand ecologisch systeem dat in een dynamisch evenwicht is kan zijn.




```
 100 REM programma 40
     roofdier-prooidier populatie
 110 CLS
 120 INPUT "beginpopulatie prooi
dieren (200)   ";x
 130 INPUT "beginpopulatie roofd
ieren (20)   ";y
 140 INPUT "groeifactor prooidie
ren (.3)   ";a
 150 INPUT "afnamefactor prooidi
eren (.01)   ";b
 160 INPUT "groeifactor roofdier
en (.002)   ";c
 170 INPUT "afnamefactor roofdie
ren (.5)   ";d
 180 CLS
 185 PLOT 0,175: DRAW 0,-175: DR
AW 255,0
 190 LET k=0.3: LET h=0.5: LET v
=0
 200 FOR u=0 TO 247 STEP 8
 210   LET x1=u
 220   LET xp=x+(a*x-b*x*y)
 230   LET yr=y+(c*x*y-d*y)
 240   REM pop.prooidieren tekene
n
 250   LET y1=INT (v+k*x+h)
 260   LET x2=x1+8: LET y2=INT (v
+k*xp+h)
 270   IF y2<0 OR y2>175 THEN GO
TO 360
 280   PLOT x1,y1: DRAW x2-x1,y2-
y1
 290   REM pop. roofdieren tekene
n
 300   LET y1=INT (v+k*y+h)
 310   LET x2=x1+8: LET y2=INT (v
+k*yr+h)
 320   IF y2<0 OR y2>175 THEN GO
TO 360
 330   PLOT x1,y1: DRAW x2-x1,y2-
y1
 340   LET x=xp: LET y=yr
 350 NEXT u
 360 IF INKEY$="" THEN GO TO 360
 370 STOP
```

Appendix

In deze appendix laten we zien hoe een aantal programma's uitgebreid kunnen worden om leuke effecten te bereiken. Ook geven we nog een aantal tekeningen die u met de programma's kunt maken. Tot slot volgt dan nog een betere versie van het bolprogramma 25.

Programma 4

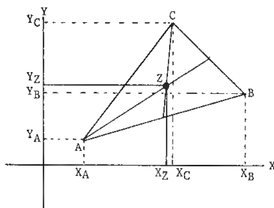
In de tekening, die programma 4 oplevert, zien we dat tussen een zeshoek en de daarbinnenliggende zeshoek zes driehoeken ontstaan. We gaan deze driehoeken om en om 'inkleuren'. We doen dit niet alleen voor de driehoeken tussen de buitenste zeshoek en de eerste ingeschreven zeshoek, maar ook bij de driehoeken tussen de eerste ingeschreven zeshoek en de tweede ingeschreven zeshoek, voor de driehoeken tussen de tweede ingeschreven zeshoek en de derde ingeschreven zeshoek, enzovoorts. Er ontstaan dan drie 'zwarte' spiraalvlakken.

Als we een vlak, in dit geval een driehoek, dat geheel omsloten wordt door lijnen willen inkleuren, moeten we in dat vlak een punt lokaliseren. De schermcoördinaten van dat punt gebruiken we dan in de 'paint-routine' om het vlak waarin dat punt ligt te kleuren (zwart of een andere kleur). Omdat er een groot aantal driehoeken gekleurd moeten worden, moeten we ook een groot aantal coördinaten van punten in die driehoeken berekenen. Deze punten moeten op een systematische manier in de FOR-lus (regels 220-350, p.115) bepaald worden. Als we in het X-Y-vlak een driehoek ABC tekenen waarvan de coördinaten van de hoekpunten $(X_A,Y_A);(X_B,Y_B)$ en (X_C,Y_C) zijn (zie de tekening op p.110), dan weten we uit de meetkunde dat de coördinaten van het zwaartepunt (Z) van de driehoek gelijk zijn aan

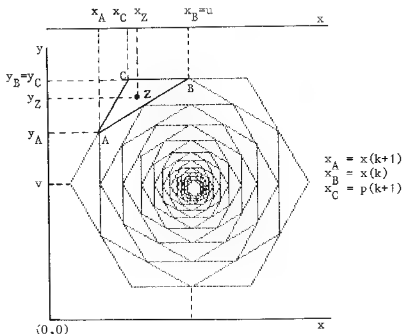
$$\left(\frac{X_A + X_B + X_C}{3}\ ,\ \frac{Y_A + Y_B + Y_C}{3}\right).$$

Het zwaartepunt van een driehoek is het snijpunt van de drie lijnen die elk een hoekpunt met het midden van de daartegenoverliggende zijde verbinden. In de tekening is Z het zwaartepunt en er geldt dus dat

$$X_Z = \frac{X_A + X_B + X_C}{3} \quad \text{en} \quad Y_Z = \frac{Y_A + Y_B + Y_C}{3}\ .$$



We kiezen in ons programma steeds het zwaartepunt van de driehoek als inwendig punt. Om de coördinaten hiervan te kunnen berekenen moeten we dus de drie x- en de drie y-coördinaten van de hoekpunten weten. We hebben in de figuur op pagina 111, die door programma 4 getekend wordt, één driehoekje als voorbeeld genomen. Van dit driehoekje zijn de hoekpunten A en B twee hoekpunten van de ingeschreven zeshoek, terwijl hoekpunt C een hoekpunt is van de daarvoor getekende (in ons voorbeeld de buitenste) zeshoek. We moeten dus steeds van twee zeshoeken de coördinaten van alle zes hoekpunten ter beschikking hebben. In het onderstaande programma hebben we, om de coördinaten van de hoekpunten van de zojuist getekende zeshoek te kunnen onthouden voordat de nieuwe berekend worden, twee arrays extra opgenomen, en wel p(7) en q(7). In regel 325 maken we p(j) en q(j) gelijk aan de coördinaten van de zojuist getekende zeshoek (x(j) en y(j)), vlak voordat x(j) en y(j) gelijk worden gemaakt aan de coördinaten van de hoekpunten van de volgende zeshoek (a(j) en b(j)) in regel 330). Zie p.115.



We zien vervolgens in de regels 263 en 264 hoe we de schermcoördinaten van het zwaartepunt berekenen uit de twee hoekpunten van de nieuwe zeshoek x(k), x(k+1) en y(k) en y(k+1) en één hoekpunt van de daarvoor getekende zeshoek (p(k+1) en q(k+1)). In de FOR-NEXT-lus van regel 262 nemen we de stapgrootte 2 om steeds één driehoek over te slaan. Natuurlijk hoeven we dit inkleuren pas te doen als de tweede zeshoek getekend is, vandaar de IF n=1 THEN GO TO 270 in regel 261. Het inkleuren gebeurt tenslotte in regel 265 en 266 met de opdrachten PLOT x,y en RANDOMIZE USR 64800. We leggen dit hierna uit. Na deze uitleg drukken we af hoe de 'gekleurde' zeshoeken eruitzien, gevolgd door het hiervoor benodigde, aangepaste, programma 4.

We hebben in het blad *Your computer* van oktober 1983 een routine gevonden voor het kleuren van een vlak, dat geheel door lijnen omsloten wordt. Willen we zo'n vlak kleuren dan moeten we de grafische cursor met een PLOT-opdracht naar de coördinaten van een punt ergens binnenin dat vlak brengen. In programma 4 hebben we steeds als 'inwendig' punt van een te kleuren (op te vullen) driehoek het zwaartepunt van zo'n driehoek genomen. Met PLOT x,y (regel 265 in het aangepaste programma 4) brengen we de cursor dus naar een punt binnenin zo'n driehoek. Met RANDOMIZE USR 64800 roepen we een machinetaalprogramma aan dat we hiervoor speciaal in het geheugen gePOKEd hebben. Deze machinetaalroutine

kleurt het vlak waarin het punt (x,y) ligt in de dan geldende afdrukkleur. In ons programma is dit zwart, maar wellicht kunt u deze kleur veranderen of zelfs drie spiralen van verschillende kleur maken.

Wilt u in uw grafische programma's ook vlakken vullen, dan zult u eerst de machinecode uit het onderstaande BASIC-programma in het geheugen moeten 'laden'. U kunt dit doen door het onderstaande BASIC-programma in te lezen en te draaien. Als alles goed gaat 'vernietigt' het programma zichzelf en kunt u de, door dit programma in het geheugen, gePOKEte machinecode SAVEn. Als u vlakken wilt kleuren moet u eerst deze machinecode LOADen (zie p.114). U kunt dan altijd, als u dat wilt, aan een programma een PLOT- en een RANDOMIZE USR-opdracht toevoegen als u iets wilt kleuren.

```
  1 REM paint routine
 10 DATA "F3C3AA7DFA6400C53EAF"
 20 DATA "9030043E01C1C9E5CDCE"
 30 DATA "22CDA22DE179C1C9C53E"
 40 DATA "AF9030043E00C1C94759"
 50 DATA "480600CB21CB10CB21CB"
 60 DATA "10CB21CB10CB21CB10CB"
 70 DATA "21CB107BCB3BCB3BCB3B"
 80 DATA "16002A047D1909CB23CB"
 90 DATA "23CB23C17993C801C947"
100 DATA "7ECB2710FC38033E00C9"
110 DATA "3E01C9473E00371F10FD"
120 DATA "B677C9C5E5D5CD1C7DFE"
130 DATA "002004D1E1C1C9CD597D"
140 DATA "D1E1C1C9C5E5D5CD1C7D"
150 DATA "FE002004D1E1C1C9CD67"
160 DATA "7DD1E1C1C9C5F53EAF90"
170 DATA "3805E5CDDF22E1F1C1C9"
180 DATA "2A047D01001636002308"
190 DATA "7881FE0020F62A7D5CE5"
200 DATA "444D0CCD077DFE00200A"
210 DATA "CD9B7DCD867D18F00000"
220 DATA "E1444D0DCD077DFE0020"
230 DATA "08CD9B7DCD867D18F03E"
240 DATA "0032815C2A047D0100AF"
250 DATA "7EFE00C4597E2379C608"
260 DATA "FE002001054F3EFFB828"
270 DATA "7A18E9CD717DFE00C804"
280 DATA "CD077DFE00CC6D7E050C"
290 DATA "CD077DFE00CC6D7E0D05"
300 DATA "CD077DFE00CC6D7E040D"
310 DATA "CD077DFE00CC6D7E0CC9"
320 DATA "E5112000237ECB7FE1C8"
330 DATA "E5287ECB47E1C8E5197E"
340 DATA "FEFFE1C0E5A7ED527EFE"
350 DATA "FFE1C0D1C9CD367EC51E"
360 DATA "00D5CD077ED10C1C3E08"
370 DATA "BB20F4C1C9CD717DFE01"
380 DATA "C8CD867DCD9B7D3E0132"
390 DATA "815CC93A815CFE00CACC"
400 DATA "7E3E0032815C2A047D11"
410 DATA "FF151901FF007EFE00C4"
420 DATA "AD7E2B79D608FEFF2001"
430 DATA "044F3EB0B8281618E9CD"
440 DATA "367EC51E00D5CD077ED1"
450 DATA "0D1C3E08BB20F4C1C93A"
460 DATA "815CFE00CACC7EC3E57D"
470 DATA "2A047D1100000100167E"
480 DATA "C50608CB2730011310F9"
490 DATA "C10B2378B1FE0020EC42"
500 DATA "4BFBC900000000000000"
510 CLEAR 31999: LET A=32000: L
```

```
ET F=0: DIM A(50)
 520 DATA 7377,6807,8313,5421,63
42,6660,6658,4971,6718,4636,4566
,9558,8293,9034,6595,9403,10142,
1364,7284,4029,5218,4938,7059,37
22,5694,5383,6622
 530 DATA 4770,4772,4771,6723,77
52,7927,10343,7611,3899,6077,534
7,7981,3266,6525,6272,6705,7010,
7055,9062,1958,4321,6249,1180,77
33262
 540 RESTORE 520: LET C=0: FOR I
=1 TO 50: READ A(I): LET C=C+A(I
)*I: NEXT I: READ V: IF C<>V THE
N PRINT "ERROR IN LINES 520 OR 5
30": STOP
 600 FOR I=10 TO 500 STEP 10: RE
STORE I: LET C=0: READ A$
 605 PRINT I
 610 FOR N=1 TO 19 STEP 2: LET L
=CODE A$(N+1): LET H=CODE A$(N)
 620 LET V=(L-48-(L>57)*7)+(H-48
-(H>57)*7)*16: LET C=C+V*((N+1)/
2)
 630 POKE A,V: LET A=A+1: NEXT N
: IF C<>A(I/10) THEN PRINT "ERRO
R IN LINE ";I: LET F=1
 640 NEXT I
 650 IF F=0 THEN PRINT "AL OK. █
YE": BEEP 1,0: POKE 23627,PEEK 2
3635: POKE 23628,PEEK 23636: CLE
AR 25800
 660 PRINT ''"CORRECT MISTAKES A
ND RERUN."
```

Dit programma doet het zowel op een 16K als op een 48K Spectrum. Hebt u een 48K Spectrum, dan kunt u de machinecode naar een hoger adres verplaatsen. Hiervoor moet het volgende programma gedraaid worden. Denk erom: dit kan alléén voor de 48K Spectrum, maar het is niet noodzakelijk om de programma's samen met de paint-routine in dit boek te kunnen draaien.

```
   1 REM relocation program
   2 REM to convert paint to 48k
   3 REM
   9 LET a=32600
  10 FOR i=32000 TO 32500
  15 LET v=PEEK (i+1)+256*PEEK (
i+2)
  20 IF (PEEK i=42 OR PEEK i=194
 OR PEEK i=195 OR PEEK i=196 OR
PEEK i=202 OR PEEK i=204 OR PEEK
 i=205 OR PEEK i=210 OR PEEK i=2
12 OR PEEK i=218 OR PEEK i=220)
AND v>32000 AND v<32500 THEN POK
E i+a,PEEK i: POKE i+1+a,(v+a)-2
56*INT ((v+a)/256): POKE i+a+2,I
NT ((v+a)/256): LET i=i+2: GO TO
 40
  30 POKE i+a,PEEK i
  40 NEXT i: POKE 64604,3: POKE
64605,229: CLEAR 58600
```

Als u het programma met de paint-routine ingevoerd hebt en draait, zal het zichzelf vernietigen als het geen fouten in de DATAregels 10 t/m 500 gevonden heeft. Het programma staat dan niet meer in het geheugen maar het machinetaalprogramma, dat gecodeerd in de DATAregels 10 t/m 500 staat, zit nog wel in het geheugen. U kunt vervolgens dit machinetaalprogramma SAVEn met de opdracht:

```
SAVE "FILL" CODE 32000,500
```

Hebt u op uw 48K Spectrum het 'relocation program' gedraaid, dan moet u de machinecode SAVEn met:

```
SAVE "FILL" CODE 64800,500
```

Wilt u de paint-routine gebruiken dan hoeft u alleen maar het machinetaalprogramma in te lezen. U doet dit met:

```
CLEAR 25800 : LOAD "FILL" CODE
```

Op uw 48K Spectrum kunt u gebruiken:

```
CLEAR 58600 : LOAD "FILL" CODE
```

Mocht u de paint-routine in uw programma's gaan gebruiken dan moet u aan het begin van zo'n programma opnemen:

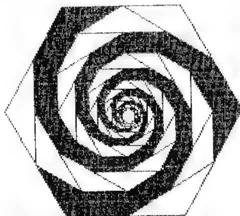
`CLEAR 25800` en bij 48K: `CLEAR 58600`

We zien dit in regel 115 van het onderstaande programma 4, waarin we (op een 48K Spectrum) van de paint-routine gebruik maken. Het aanroepen van de paint-routine gebeurt dus door eerst een inwendig punt (x,y) van het te kleuren vlak te bepalen en met PLOT x,y de cursor ernaartoe te brengen; daarna kan de paint-routine worden aangeroepen. Dit kan met

`RANDOMIZE USR 32000` in een 16K Spectrum en
`RANDOMIZE USR 64800` als u een 48K Spectrum hebt.

U kunt de routine ook aanroepen met LET A=USR 32000 respectievelijk LET A=USR 64800.

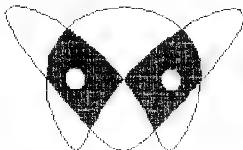
De paint-routine volgt de eventueel gegeven INK-, BRIGHT- en FLASH-opdrachten. Zorg dat het te kleuren vlak geheel omsloten is door lijnen (er mag geen pixel in ontbreken) anders zal de routine het hele scherm vullen!
In het onderstaande programma 4 ziet u hoe de routine wordt aangeroepen. Eerst geven we echter het resultaat van het programma.



```
100 REM programma 4
        ingeschreven zeshoeken
110 CLS
115 CLEAR 58600
120 DIM x(7) : DIM y(7)
130 DIM a(7) : DIM b(7)
135 DIM p(7): DIM q(7)
140 LET u=128 : LET v=88
150 LET r=88 : LET h=0.5
160 LET w=60*PI/180
170 FOR j=1 TO 7
180  LET w1=j*w
190  LET x(j)=INT (u+r*COS (w1)
+h)
200  LET y(j)=INT (v+r*SIN (w1)
+h)
210 NEXT j
220 FOR n=1 TO 20
230  FOR j=1 TO 6
240   PLOT x(j),y(j)
250   DRAW x(j+1)-x(j),y(j+1)-y
(j)
260  NEXT j
261  IF n=1 THEN GO TO 270
262  FOR k=2 TO 6 STEP 2
263   LET x=INT ((x(k)+x(k+1)+p
(k+1))/3)
264   LET y=INT ((y(k)+y(k+1)+q
(k+1))/3)
265   PLOT x,y
266   RANDOMIZE USR 64600
267  NEXT k
270  FOR k=1 TO 6
280   LET a(k)=INT ((x(k)+x(k+1
))/2+h)
290   LET b(k)=INT ((y(k)+y(k+1
))/2+h)
300  NEXT k
310  LET a(7)=a(1) : LET b(7)=b
(1)
320  FOR j=1 TO 7
325   LET p(j)=x(j): LET q(j)=y
(j)
330   LET x(j)=a(j) : LET y(j)=
b(j)
340  NEXT j
350 NEXT n
360 IF INKEY$="" THEN GO TO 360
370 STOP
```

Programma 19

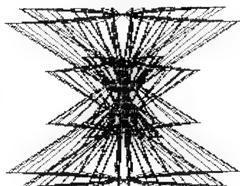
We kunnen van de vliegekop, die programma 19 tekent, een 'echte' vliegekop met ogen maken. Hiertoe tekenen we twee ogen (cirkels) en kleuren het vlak dat door de cirkels en de daarbuitenliggende lijnen begrensd wordt. Het resultaat en het gewijzigde programma 19 staan hieronder.
In de regels 221 en 222 tekenen we de twee ogen met de CIRCLE-opdracht. In de regels 223, 224 en 225, 226 kleuren we de vlakjes rond de ogen met de machinetaalroutine die we in het geheugen gePOKEd hebben, net zoals we dat bij het vorige programma gedaan hebben.
Probeer de ogen en het vlak rond de ogen een kleur te geven door de INK-opdracht te gebruiken. Wellicht kunt u de ogen ook laten schitteren met de FLASH-opdracht?



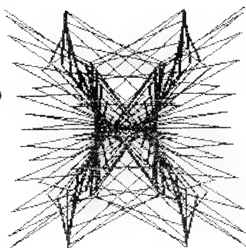
```
 100 REM programma 19 vliegenkop
 105 CLEAR 58600
 110 CLS
 120 LET u=128: LET v=87
 130 LET h=0.5: LET k=30: LET rd
=PI/180
 140 FOR w=90 TO 450 STEP 3
 150  LET t=w*rd: GO SUB 1000
 160  LET x= INT (u+x+h)
 170  LET y=INT (v+y+h)
 180  IF w=90 THEN LET x1=x: LET
y1=y: GO TO 220
 190  LET x2=x: LET y2=y
 200  PLOT x1,y1: DRAW x2-x1,y2-
y1
 210  LET x1=x2: LET y1=y2
 220 NEXT [illegible]
 221 CIRCLE 168,90,10
 222 CIRCLE 88,90,10
 223 PLOT 180,[illegible]
 224 RANDOMIZE USR 64800
 225 PLOT 78,90
 226 RANDOMIZE USR 64800
 230 IF  INKEY$="" THEN GO TO 23
0
 240 STOP
 250:
1000 LET x=k*SIN (2*t)*(2.5+COS
(3*t))
1010 LET y=k*2*COS (3*t)
1020 RETURN
```

Programma 21

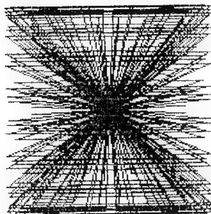
Allereerst geven we een paar voorbeelden van wat je zoal met programma 21 kunt tekenen. Je zou dit wellicht 'computerkunst' mogen noemen.



a=1/b=-100/c=-200



a=-300
b=-1
c=299



a=3/b=100/c=200

Tekst op het grafische scherm

Het zou leuk zijn op het grafische scherm de waarden voor a, b en c af te drukken naast de tekening. Bevalt zo'n tekening, dan kunnen we direct de waarden van a, b en c noteren. Op de ZX Spectrum is dit helemaal niets bijzonders. We kunnen zonder veel moeite tekst en graphics op het scherm afbeelden; dat hebben we in programma 39 immers al gezien. We volstaan daarom met het geven van programma 21, waarin PRINT-opdrachten zijn opgenomen voor het afdrukken van de waarde van a, b en c op het 'grafische' scherm.

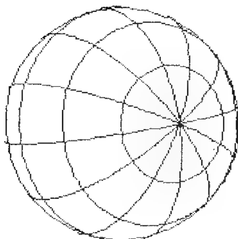
```
 100 REM programma 21
         symmetrische krommen
 110 CLS
 120 INPUT "Toets a,b,c in  ";a,
b,c
 130 LET u=128: LET v=88
 140 LET h=0.5: LET rd=PI/180: L
ET k=87
 150 CLS
 160 FOR w=0 TO 360
 170  LET t=w*rd: LET r=k*SIN (c
*t)
 180  LET x2=INT (u+r*COS (a*t)+
h)
 190  LET y2=INT (v+r*SIN (b*t)+
h)
 200  IF w=0 THEN LET x1=x2: LET
 y1=y2: GO TO 230
 210  PLOT x1,y1: DRAW x2-x1,y2-
y1
 220  LET x1=x2: LET y1=y2
 230 NEXT w
 231 PRINT AT 19,1;"a: ";a
 232 PRINT AT 20,1;"b: ";b
 233 PRINT AT 21,1;"c: ";c
 240 IF INKEY$="" THEN GO TO 240
 250 STOP
```

Programma 25

De auteur Marcel Sutter heeft onlangs in het tijdschrift *Mikro+Klein-computer* een verbeterde versie van het 'bolprogramma' 25 gepubliceerd. We laten dit nu zien. Het programma tekent bollen in allerlei standen en met verborgen lijnen. De draaihoeken om de x-, y- en z-as alsmede de 'afstand' tussen de breedte- en lengtelijnen kunnen worden ingevoerd. Eerst volgt het programma, daarna enkele voorbeelden.

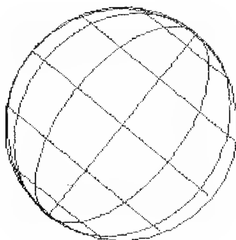
```
  50 REM bol met hidden lines en draaing
 100 CLS
 105
 110 PRINT "BOL MET LENGTE EN BREEDTECIRKELS"
 120 PRINT "------------------------------"
 130 PRINT : PRINT
 200 INPUT "DRAAIHOEK OM X-AS: ";a: PRINT
 210 INPUT "DRAAIHOEK OM Y-AS: ";b: PRINT
 220 INPUT "DRAAIHOEK OM Z-AS: ";c: PRINT
 225 INPUT "AFSTAND LIJNEN (10-45): ";d
 230 LET u=128: LET v=87: LET r=87: LET bm=PI/180: LET h=0.5
 240 LET s1=SIN (a*bm): LET s2=SIN (b*bm): LET s3=SIN (c*bm)
 250 LET c1=COS (a*bm): LET c2=COS (b*bm): LET c3=COS (c*bm)
 255
 260 REM ROTATIEMATRIX BEREKENEN
 270 LET ax=c2*c3: LET ay=-c2*s3: LET az=s2
 280 LET bx=c1*s3+s1*s2*c3
 290 LET by=c1*c3-s1*s2*s3: LET bz=-s1*c2
 300 LET cx=s1*s3-c1*s2*c3
 310 LET cy=s1*c3+c1*s2*s3: LET cz=c1*c2
 395
 400 REM OMTREK CIRKEL TEKENEN
 405 CLS
 410 CIRCLE u,v,r
 415
 500 REM LENGTECIRKELS TEKENEN
 510 FOR l=0 TO 180-d STEP d
 515  LET f1=0
 520  FOR p=0 TO 360 STEP 5
 530   GO SUB 1000: REM XX,YY,ZZ BEREKENEN
 540   IF yy>0 THEN LET f2=0: LET f1=0: GO TO 580
 550   LET xb=INT (u+xx+h): LET yb=INT (v+zz+h): LET f2=1
 560   IF f1=0 THEN LET x1=xb: LET y1=yb: LET f1=1: GO TO 580
 570   PLOT x1,y1: DRAW xb-x1,yb-y1: LET x1=xb: LET y1=yb: LET f1=f2
 580  NEXT p
 590 NEXT l
```

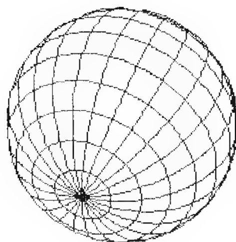
```
 595
 600 REM BREEDTECIRKELS TEKENEN
 610 FOR p=-90+d TO 90-d STEP d
 615  LET f1=0
 620  FOR l=0 TO 360 STEP 5
 630   GO SUB 1000: REM XX,YY,ZZ
 BEREKENEN
 640   IF yy>0 THEN LET f2=0: LE
T f1=0: GO TO 680
 650   LET xb=INT (u+xx+h): LET
yb=INT (v+zz+h): LET f2=1
 660   IF f1=0 THEN LET x1=xb: L
ET y1=yb: LET f1=1: GO TO 680
 670   PLOT x1,y1: DRAW xb-x1,yb
-y1: LET x1=xb: LET y1=yb: LET f
1=f2
 680  NEXT l
 690 NEXT p
 695
 700 IF INKEY$="" THEN GO TO 700
 710 STOP
 715
 999 REM BOLCOORDINATEN -> CARTE
SISCHE COORDINATEN
1000 LET x=r*COS (p*bm)*COS (l*b
m)
1010 LET y=r*COS (p*bm)*SIN (l*b
m)
1020 LET z=r*SIN (p*bm)
1025
1030 REM P(X,Y,Z) DRAAIEN TOT P(
XX,YY,ZZ)
1040 LET xx=ax*x+ay*y+az*z
1050 LET yy=bx*x+by*y+bz*z
1060 LET zz=cx*x+cy*y+cz*z
1100 RETURN
```



draaihoek om x-as: 90
draaihoek om y-as: 30
draaihoek om z-as: 40
afstand lijnen : 30

draaihoek om x-as: 0
draaihoek om y-as: 40
draaihoek om z-as: 60
afstand lijnen : 30





draaihoek om x-as: -50
draaihoek om y-as: 20
draaihoek om z-as: 60
afstand lijnen : 15

Programma 36

We gaan de kaart van Zwitserland wat verfraaien. Eerst geven we het gewijzigde programma.

```
 100 REM programma 36
          kaart van Zwitserland
 105 CLEAR 58600
 110 CLS
 120 LET k=1.3: LET h=0.5: LET u
=15: LET v=175
 130 READ x,y
 140 LET x1=INT (u+k*x+h): LET y
1=INT (v-k*y+h)
 150  LET x2=INT (u+k*x+h): LET
y2=INT (v-k*y+h)
 160  PLOT x1,175-y1: DRAW x2-x1
,y1-y2
 170  LET x1=x2: LET y1=y2
 180  READ x,y
 190 IF x<>0 THEN GO TO 150
 195 LET a$=INKEY$
 200 IF a$="" THEN GO TO 195
 201 IF a$="p" THEN PLOT 128,88:
 RANDOMIZE USR 64600: LET a$=INK
EY$: GO TO 200
 202 IF a$="v" THEN LET u=128: L
ET v=88: LET k=1: GO TO 130
 203 IF a$="k" THEN PLOT 128,96:
 RANDOMIZE USR 64600: LET a$=INK
EY$: GO TO 200
 210 STOP
 220:
 300 DATA 69,108,71,107,70,104,7
5,104,75,104,76,106
 310 DATA 80,107,81,104,86,105,9
1,108,94,107
 320 DATA 101,106,100,108,105,10
8,108,110,101,110
 330 DATA 98,112,102,117,108,118
,112,112,114,115
 340 DATA 118,110,128,110,139,10
2,145,103,146,95
 350 DATA 142,85,144,78,154,77,1
54,77,154,72,163,67
 360 DATA 168,68,173,76,177,73,1
74,59,177,56
 370 DATA 177,52,171,51,167,56,1
61,50,165,43
 380 DATA 166,34,162,34,157,42,1
43,36,139,48
 390 DATA 136,45,133,48,132,40,1
22,23,125,15
 400 DATA 122,11,119,12,114,20,1
16,25,100,35
 410 DATA 102,45,94,42,88,35,90,
28,79,15
 420 DATA 75,15,66,19,60,14,52,1
2,48,13
 430 DATA 37,29,39,36,37,40,39,4
3,29,45
 440 DATA 16,38,18,33,13,29,6,28
,6,32
 450 DATA 11,34,13,40,10,45,12,5
3,25,63
 460 DATA 26,73,30,73,46,94,42,9
4,46,102
```

```
470 DATA 54,102,53,99,61,98,63,
102,69,108
480 DATA 0,0
490 DATA -40,10,-10,10,-10,40,1
0,40
500 DATA 10,10,40,10,40,-10,10,
-10
510 DATA 10,-40,-10,-40,-10,-10
,-40
520 DATA -10,-40,10,0,0
```



Het verschil met het programma op p.97 is dat na regel 190 een aantal opdrachten is toegevoegd. Door het intikken van een 'p', 'v' of 'k' gebeuren er nu verschillende dingen. Draai het programma en kijk of u ook de hierbovenstaande illustratie op uw scherm kunt krijgen.

Het zal duidelijk zijn dat de in dit boek gegeven programma's onbeperkt uitgebreid kunnen worden. Ze kunnen vast ook verbeterd worden; de invoerwaarden zouden bijvoorbeeld gecontroleerd kunnen worden, zodat het programma niet ergens halverwege afbreekt door een foutieve schermcoördinaat. We hebben dat allemaal niet gedaan, omdat we de programma's klein wilden houden, waardoor alle aandacht op de tekentechniek gericht blijft.

ACADEMIC SERVICE INFORMATICA UITGAVEN

AUTOMATISERING EN COMPUTERS

Computers en onze samenleving - M.A. Arbib
Computers in de negentiger jaren - G.L. Simons
De informatiemaatschappij - Jan Everink
Basiskennis informatieverwerking - Jan Everink
AIV, Automatisering van de informatieverzorging - Th.J.G. Derksen en H.W. Crins
Organisatie, informatie en computers - D.M. Kroenke
De Viewdata revolutie - S. Fedida en R. Malik

MICROCOMPUTERS

Microcomputers thuis en op school - K.P. Goldberg en R.D. Sherwood
Bouw zelf een Expertsysteem in BASIC - C. Naylor
Programmeercursus Microsoft BASIC - Nok van Veen
Werken met bestanden in BASIC - L. Finkel en J.R. Brown
40 Grafische programma's voor de Commodore 64 - M. Sutter
Doe het-zelf programma's op de Commodore 64 - D. Kreutner
Programmeercursus BASIC op de Commodore 64 - Nok van Veen
TRS-80 BASIC - Bob Albrecht e.a.
TRS-80 BASIC voor gevorderden - Don Inman e.a.
Exidy sorcerer en BASIC - Nok van Veen e.a.
40 Grafische programma's voor de Electron en BBC - M. Sutter
Het Electron en BBC Micro boek - Jim McGregor en Alan Watt
Ontdek de ZX Spectrum - Tim Hartnell
Werken met bestanden op de Apple - L. Finkel en J.R. Brown
Programmeercursus Applesoft BASIC - Nok van Veen
40 Grafische programma's voor de Apple II,IIe,IIc - M. Sutter
40 Grafische programma's in MSX BASIC - M. Sutter
Programmeercursus MSX BASIC - Nok van Veen

MICROPROCESSORS EN ASSEMBLEERTALEN

Procescomputers, basisbegrippen - dr.ir. J.E. Rooda en ir. W.C. Boot
Cursus Z-80 assembleertaal - Roger Hutty
6502 Assembleertaal en machinecode voor beginners - A.P. Stephenson

BESTURINGSSYSTEMEN

Inleiding besturingssystemen - A.M. Lister
Systeemprogrammatuur en software-ontwikkeling voor microcomputers - E. Verhulst
Bedrijfssystemen - EIT-serie, deel 4
CP/M het operating system voor microcomputers - J.N. Fernandez en R. Ashley
CP/M 86 - Nok van Veen
CP/M voor gevorderden - A. Clarke e.a.
PC DOS, het besturingssysteem van de IBM PC - R. Ashley en J.N. Fernandez
MS/DOS, het besturingssysteem voor 16 bit microcomputers - R. Ashley en J.N. Fernandez
UNIX, het standaard operating system - G.J.M. Austen en H.J. Thomassen
Werken met UNIX - Brian W. Kernighan en Rob Pike

PERSONAL COMPUTERS

Het werken met bestanden op de IBM PC - L. Finkel en J.R. Brown
De IBM PC en zijn toepassingen - Laurence Press
40 Grafische programma's voor de IBM PC - M. Sutter
Werken met VisiCalc - C. Klitzner en M.J. Plociak
Multiplan, een hulpmiddel bij de bedrijfsvoering - D.F. Cobb e.a.
Multiplan diskettes
Werken met Lotus 1-2-3 - D. Cobb en G. LeBlond

PROGRAMMEREN

Een methode van programmeren - prof.dr. Edsger W. Dijkstra en ir. W.H.J. Feijen
Programmeren, het ontwerpen van algoritmen (met Pascal) - ir. J.J. van Amstel
Inleiding tot het programmeren, deel 1 - ir. J.J. van Amstel
Inleiding tot het programmeren, deel 2 - ir. J.J. van Amstel
Programmeren, deel 2: van analyse tot algoritme - prof.drs. C. Bron
Inleiding programmeren en programmeertechnieken - EIT-serie, deel 1
Het Groot Pascal Spreuken Boek - H.F. Ledgard e.a.
JSP - Jackson Struktureel Programmeren - Henk Jansen
JSP Uitwerkingenboek - Henk Jansen

PROGRAMMEERTALEN

Aspecten van programmeertalen – ir. J.J. van Amstel en ir. J.A.A.M. Poirters
Programmeertalen, een inleiding – ir. J.J. van Amstel e.a.
BASIC – EIT-serie, deel 3
Cursus BASIC, een practicum-handleiding voor BASIC op de PRIME – ir. R. Bloothoofd e.a.
Cursus Pascal – prof.dr. A. van der Sluis en drs. C.A.C. Görts
Cursus eenvoudig Pascal – prof.dr. A. van der Sluis en drs. C.A.C. Görts
Inleiding programmeren in Pascal – C. van de Wijgaart
Systeemontwikkeling met Ada – Grady Booch
Cursus COBOL – A. Parkin
Cursus FORTRAN 77 – J.N.P. Hume en R.C. Holt
Aanvulling cursus FORTRAN 77 voor PRIME-computers – ing. J.M. den Haan
De programmeertaal C – ir. L. Ammeraal
Flitsend Forth – Alan Winfield
Programmeren in LISP – prof.dr. L.L. Steels

GEGEVENSSTRUCTUREN EN BESTANDSORGANISATIE

Informatiestructuren, bestandsorganisatie en bestandsontwerp – EIT-serie, deel 5
Programmeren, het ontwerpen van datastructuren en algoritmen – ir. J.J. van Amstel e.a.
Bestandsorganisatie – prof.dr. R.J. Lunbeck en drs. F. Remmen

DATABASE EN GEGEVENSANALYSE

Database, een inleiding – C.J. Date
Databases – drs. F. Remmen
Gegevensanalyse – R.P. Langerhorst

INFORMATIE-ANALYSE EN SYSTEEMONTWERP

Effectieve toepassingen van computers – M. Peltu
Voorbereiding van computertoepassingen – prof.dr. A.B. Frielink
Systeemontwikkeling volgens SDM – H.B. Eilers
Samenvatting SDM – Pandata
Informatie-analyse volgens NIAM – J.J.V.R. Wintraecken
Evaluation of methods and techniques for the analysis, design and implementation of information systems – ed. J. Blank en M.J. Krijger
Inleiding systeemanalyse, systeemontwerp – W.S. Davis
Systeemontwikkeling Zonder Zorgen – Paul T. Ward
Het ontwerpen van interactieve toepassingen en computernetwerken – J.A. Scheltens
EDP Audit – prof.dr. C. de Backer
Prototyping, een instrument voor systeemontwerpers – ed. T. Hoenderkamp en H.G. Sol
Simulatie, een moderne methode van onderzoek – drs. S.K. Boersma en ir. T. Hoenderkamp

EXPERT SYSTEMEN EN KUNSTMATIGE INTELLIGENTIE

Computerschaak – H.J. van den Herik
Expert systemen – Henk de Swaan Arons en Peter van Lith

THEORETISCHE INFORMATICA EN SYSTEEMPROGRAMMATUUR

Informatica, een theoretische inleiding – dr. L.P.J. Groenewegen en prof.dr. A. Ollongren
Systeemprogrammatuur – drs. H. Alblas
Vertalerbouw – H. Alblas e.a.

AANVERWANTE ONDERWERPEN EN OVERIGE TITELS

Lineaire programmering als hulpmiddel bij de besluitvorming – prof.dr. S.W. Douma
Inleiding programmeren – prof.dr. R.J. Lunbeck
Analyse van informatiebehoeften en de inhoudsbeschrijving van een databank – prof.dr. P.G. Bosch en ir. H.M. Heemskerk
Gegevensstructuren – R. Engmann e.a.
Cases en Uitwerkingenboek bij Cases – prof.dr. P.G. Bosch en H.A. te Rijdt
De tekstmachine – dr. M. Boot en drs. H. Koppelaar
Abstracte automaten en grammatica's – prof.dr. A. Ollongren en ir. Th.P. van der Weide
Onderneming en overheid in systeem-dynamisch perspectief – red. A.F.G. Hanken e.a.
Simulatie en sociale systemen – red. J.L.A. Geurts en J.H.L. Oud
Struktuur en stijl in COBOL – ir. E. Dürr en dr.ir. F. Mulder
Cursus ALGOL 60 – prof.dr. A. van der Sluis en drs. C.A.C. Görts



Over de inhoud van dit boek

Dit boek bevat 40 grafische programma's voor het programmeren met hoge resolutie in ZX Spectrum BASIC. Dit houdt in dat het scherm van uw ZX Spectrum computer verdeeld wordt in 256 puntjes horizontaal en 176 puntjes verticaal. Op dit grafische scherm kunnen fraaie tekeningen gemaakt worden.

Bij bijna elk programma wordt de nodige tekst en uitleg gegeven. Daarnaast bevat het boek veel illustraties van datgene wat u op het scherm kunt verwachten. Alle programma's zijn getest op een ZX Spectrum. De programmatekst is direct vanuit het geheugen van de ZX Spectrum op een printer afgedrukt en deze afdrukken zijn rechtstreeks in het boek opgenomen.

Laat u niet afschrikken door de wat wiskundige uitleg bij de programma's. Ook voor niet-wiskundige Spectrum-bezitters is dit boek bedoeld. Het alleen maar intikken en draaien van de programma's geeft al zulke fraaie beelden, dat het bestuderen van de erachterliggende theorie geenszins noodzakelijk is, alhoewel de 'kijker' hiertoe wel geprikkeld wordt.

Naast het tekenen van allerlei eenvoudige en meer ingewikkelde functies (voor leraren en scholieren) worden de beginselen van het driedimensionaal tekenen uitgelegd. Ook de techniek van de verborgen lijnen (hidden lines) wordt gebruikt. Vijf programma's in BASIC laten zien wat voor figuren je met de taal LOGO kunt maken. Hieronder zijn ook de beroemde turtle-graphics.
Ook bevat het boek vijf educatieve toepassingsprogramma's, waarin met graphics gewerkt wordt.

In een appendix wordt een aantal programma's verfraaid en uitgebreid om de lezer de vele mogelijkheden van ZX Spectrum graphics te laten zien.
Deze appendix bevat een programma voor het inlezen van een paint routine (in machine code), die niet standaard in ZX Spectrum BASIC voorhanden is.
Met deze routine kunnen vlakken worden gevuld of ingekleurd.

De illustratie op de voorkant is gemaakt met het onderstaande programma

```
100 REM programma 1
       diagonaalweb
110 CLS
120 LET y1=0 : LET y2=175
130 FOR a=0 TO 255 STEP 36
140  FOR b=0 TO 255 STEP 36
150   PLOT a,y1
160   DRAW b-a,y2
170  NEXT b
180 NEXT a
190 IF INKEY$="" THEN GO TO 190
200 STOP
```

ISBN 90 6233 180 7

Over de inhoud van dit boek

Dit boek bevat 40 grafische programma's voor het programmeren met hoge resolutie in ZX Spectrum BASIC. Dit houdt in dat het scherm van uw ZX Spectrum computer verdeeld wordt in 256 puntjes horizontaal en 176 puntjes verticaal. Op dit grafische scherm kunnen fraaie tekeningen gemaakt worden.

Bij bijna elk programma wordt de nodige tekst en uitleg gegeven. Daarnaast bevat het boek veel illustraties van datgene wat u op het scherm kunt verwachten. Alle programma's zijn getest op een ZX Spectrum. De programmatekst is direct vanuit het geheugen van de ZX Spectrum op een printer afgedrukt en deze afdrukken zijn rechtstreeks in het boek opgenomen.

Laat u niet afschrikken door de wat wiskundige uitleg bij de programma's. Ook voor niet-wiskundige Spectrum-bezitters is dit boek bedoeld. Het alleen maar intikken en draaien van de programma's geeft al zulke fraaie beelden, dat het bestuderen van de erachterliggende theorie geenszins noodzakelijk is, alhoewel de 'kijker' hiertoe wel geprikkeld wordt.

Naast het tekenen van allerlei eenvoudige en meer ingewikkelde functies (voor leraren en scholieren) worden de beginselen van het driedimensionaal tekenen uitgelegd. Ook de techniek van de verborgen lijnen (hidden lines) wordt gebruikt. Vijf programma's in BASIC laten zien wat voor figuren je met de taal LOGO kunt maken. Hieronder zijn ook de beroemde turtle-graphics.
Ook bevat het boek vijf educatieve toepassingsprogramma's, waarin met graphics gewerkt wordt.

In een appendix wordt een aantal programma's verfraaid en uitgebreid om de lezer de vele mogelijkheden van ZX Spectrum graphics te laten zien.
Deze appendix bevat een programma voor het inlezen van een paint routine (in machine code), die niet standaard in ZX Spectrum BASIC voorhanden is.
Met deze routine kunnen vlakken worden gevuld of ingekleurd.

De illustratie op de voorkant is gemaakt met het onderstaande programma

```
100 REM programma 1
      diagonaalweb
110 CLS
120 LET y1=0 : LET y2=175
130 FOR a=0 TO 255 STEP 36
140  FOR b=0 TO 255 STEP 36
150   PLOT a,y1
160   DRAW b-a,y2
170  NEXT b
180 NEXT a
190 IF INKEY$="" THEN GO TO 190
200 STOP
```

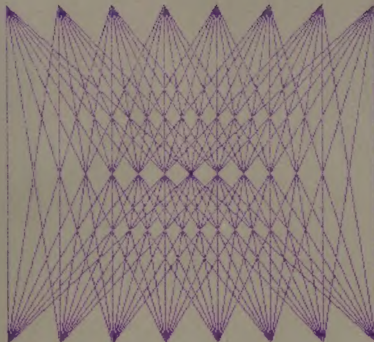
ISBN 90 6233 180 7

M. Sutter 40 grafische programma's voor de ZX Spectrum

40 grafische programma's voor de ZX Spectrum

Leer programmeren
met hoge resolutie graphics
in BASIC

M. Sutter



ACADEMIC SERVICE